Architectures for Safe Autonomy:
Provable Guarantees Across Control, Planning, and Perception

by

Devansh R. Agrawal

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Aerospace Engineering)
in the University of Michigan
2025

Doctoral Committee:

Professor Dimitra Panagou, Chair
Professor Ilya Kolmanovsky
Professor Necmiye Ozay
Professor Vasileios Tzoumas

Devansh R. Agrawal
devansh@umich.edu
ORCID iD: 0000-0002-0236-9077

ACKNOWLEDGEMENTS

This dissertation would not have been possible without a large (but countably finite) number of people that have supported me and championed me throughout the years. My PhD advisor, Prof Dimitra Panagou has had the most significant and direct impact on shaping me as an academic - she took in someone with a very limited understanding and experience with controls, and tirelessly introduced me to new ideas and references to help me learn. She has been incredibly patient and meticulous in both the idea development and the paper writing/presenting phases of research, giving me both the flexibility to explore new ideas, while still ensuring I actually submitted my papers on time (mostly). Amongst the many many other people who deserve to be mentioned here, I would in particular like to thank Mr Terence Chiew for getting me excited for physics and setting me on my path, and Prof John Ho for teaching me how research works well before I joined a PhD program.[1]

This dissertation would also not have been possible without my friends. A special shoutout goes to Laura, Brian, Shashank and Alia, who have been my family throughout gradschool. My labmates, collaborators, and all of the friends in the robotics building and at conferences have made this experience one to cherish.

Finally, I would like to thank my family for always encouraging me to learn, and to never let me doubt that this was the right path for me.

Thank you, everyone, for everything.

[1] http://phdcomics.com/comics.php?f=1431, http://phdcomics.com/comics.php?f=405, http://phdcomics.com/comics.php?f=581

TABLE OF CONTENTS

LIST OF FIGURES

FIGURE

LIST OF TABLES

TABLE

LIST OF ACRONYMS

BCH Baker-Campbell-Hausdorff

CBF Control Barrier Function

CLF Control Lyapunov Function

DCT Discrete Cosine Transform

DMP Distance Map Planner

ESDF Euclidean Signed Distance Field

FoV Field of View

FPV First Person View

GP Gaussian Process

HOCBF Higher Order Control Barrier Function

ICCBF Input-Constrained Control Barrier Function

ISS Input-to-State

KF Kalman Filter

ML Machine Learning

MPC Model Predictive Control

ODE Ordinary Differential Equation

QP Quadratic Program

RGBD RGB-Depth

RL Reinforcement Learning

RoS Rate of Spread

SDE Stochastic Differential Equation

SDF Signed Distance Field

SFC Safe Flight Corridor

SLAM Simultaneous Localization and Mapping

SOS Sum of Squares

TCAC Technical Committee on Aerospace Controls

TSDF Truncated Signed Distance Field

TSD Target Spatial Distribution

VIO Visual Inertial Odometry

VO Visual Odometry

ABSTRACT

This thesis focuses on the design of safety-critical autonomous systems - systems that must always satisfy a set of safety constraints. The primary objective is to develop a cohesive architecture for the entire autonomy stack, ensuring that, under specific and verifiable assumptions, a robot can execute its mission while maintaining safety.

Modern autonomous systems present unique challenges because their autonomy stacks are composed of interdependent modules: (1) a mission-level planning module that makes high-level decisions, (2) a perception module that processes sensor data to estimate the robot's state and the operating environment, (3) a planning module that generates a trajectory for execution, and (4) a control module that computes actuation commands. Guaranteeing safety requires a systematic approach to the design and integration of these modules.

To achieve this, we take a bottom-up approach, starting with the design of a safety-critical controller and identifying the assumptions necessary for its safe operation. These assumptions impose requirements on upstream autonomy modules, such as the planning and perception modules. We then propose methods to design or augment each module to ensure that, when composed, the entire autonomy stack maintains safety guarantees. The focus is not only on individual module correctness but making assumptions for each module that can be satisfied by upstream modules, to be able to achieve system-level guarantees.

The main contributions of this thesis include: (A) the `gatekeeper` architecture - a flexible framework for establishing rigorous safety guarantees at the planning level, (B) the development of certifiably correct perception algorithms that generate accurate obstacle maps while providing error bounds to account for odometry drift, and (C) the introduction of clarity and perceivability - concepts that quantify a robotic system's ability to gather information about its environment, considering the environment model as well as the robot's actuation and sensing capabilities.

Each contribution is supported by formal proofs and validated through simulations and hardware experiments with aerial and mobile robots.

CHAPTER 1

Introduction

> A machine is anything that reduces human effort. Anything that simplifies work, or saves time, is a machine. From a pen's nib to a pants' zip - all machines. Up and down in a second! Up, down, up, down...
>
> *Rancho*
> *3 Idiots*

This thesis puts forth a set of frameworks and methods to design safety-critical autonomous systems. A key theme of the thesis is to understand how the various modules of a modern autonomy stack should be modified and integrated such that the safety guarantees can hold across the entire autonomy stack.

We start in this chapter by studying some motivating examples of autonomy systems, both of systems in operation today, and also of systems to be developed in the future. This exercise will help us identify some recurring challenges and common bottlenecks in state-of-the-art methodologies. This will help us identify suitable abstractions that the theory presented in this thesis will ultimately address. The remainder of the chapter highlights the key contributions of the thesis and the order it is presented in. Since each chapter address a different part of the autonomy stack, the relevant literature for each chapter is presented towards the beginning of the chapter.

1.1 Motivation

Modern machines are becoming increasingly capable and relied upon. As the computational capacity of our robots increases, they have greater ability to to analyze incoming data, reason about their objectives, and decide on the best course of action. Increasingly, we are

expecting these robots to be able to perform tasks in challenging and diverse environments, where it is no longer sufficient to simply define a set of primitive rules that constrain the robots operating parameters - instead, we expect our robots to be able to make intelligent decisions in the face of uncertainty. Furthermore, we expect them not to fail.

The need for guarantees

Consider, for example, the lift - found in almost every building three stories or higher across the world. Although one might not consider this a particularly interesting robotic system, think about why you trust your life to a metal box in a hollow vertical shaft every time you enter a lift. History sheds some light here. The lift was invented by Elisha Otis, who demonstrated his invention at the 1854 World's Fair [29, pg. 1]:

> *"He installed a platform on guide rails on which he had himself hoisted into the air before the onlookers. When the platform had risen to its maximum height, to their horror, he severed its suspension cable. But instead of plunging fifty feet to the ground, the elevator stopped short after only a few inches of travel. "All safe, gentlemen, all safe," Otis reassured the shocked fair-goers, and then explained his newly developed safety catch."* [29, pg. 1]

Naturally, 1854 was not the first time humanity thought of the idea of a lift - the idea of using ropes to hoist things has appeared in writing since at least Archimedes [29]. Otis' novelty lie in the safety-critical component, the automatic braking system.[1]

The lift's safety mechanism was primarily mechanical.[2] Its operational principle could be justified theoretically, and verified experimentally. As our robots get more complicated and operate in more unstructured environments, it becomes more challenging to guarantee safety. For our technologies to be used, relied upon, and built further, we require safety and performance guarantees for the system.

Guarantees for Nonlinear Systems

Modern control theory has a long and successful history, with many of the foundational results having been developed in the 1960s. This is around the time when Bellman, Pon-

[1]The lift also had significant societal impact - prior to its mass adoption, the prime real estate was apartment on the first floor, not the penthouse at the top of the sky scrapper. Once available, the rich chose to reside at the top floors, away from all of the noise and pollution near the roads [29].

[2]For the curious, the safety mechanism is as follows. A leaf spring was installed in the ceiling of the lift, and kept under tension by the suspension cable. If severed, the leaf spring would flatten, and dig into the geared teeth lining the lift shaft. This would stop the elevator.

tryagin, Lyapunov,[3] Kalman, and many more developed their theories. Coupled with the introduction of digital computer systems, these theories could be deployed on an autonomous system, with many of the early applications being for space systems [104, Ch. 1].

Although many of these ideas were quickly extended to the nonlinear setting, analytical and computational limitations meant that many of these theories could only be applied to linear control systems. Coupled with the analytical tools from Laplace, Fourier, Nyquist, Bode, etc, one could analyze, and more importantly predict, the behavior the (linear) autonomous system being controlled. With the tools from robust analysis developed in the 1980s, one could design and build systems with guarantees of stability *and* robustness to disturbances. [104, Ch. 1].

The control of nonlinear systems is significantly more challenging, and the available tools are more limited. Most of the methods extend one of two foundational principles: the dynamic programming/optimal control principles along the lines of Bellman and Pontryagin (e.g. [70]), or the Lyapunov function based feedback control methods (e.g. [91]). Although both methods are developed for general nonlinear systems, they can be difficult to use. Optimal control methods scale poorly with the number of state, while finding a Lyapunov method is often challenging.

Nonetheless, since many real-world robotic systems are nonlinear, it is important for the tools we develop to be applicable for nonlinear systems, since we still require guarantees for these systems. A common paradigm (especially in the controls community) is to design controllers using a linear or linearized model, and then analyze the convergence properties of the nonlinear system with the linearized controller [91]. Another common paradigm (popular in the robotics community) is to decompose the controller into two modules, where the first uses linear/linearized models to compute trajectories, and the second module uses a nonlinear trajectory tracking controller to execute the commands.

Interfacing Modules within the Autonomy Stack

The problem becomes even more challenging when considering robots that operate in unstructured environments. For instance, consider a drone flying through a set of previously unmapped obstacles. In this case, the drone must sense the world through onboard sensors, identify the positions of the obstacles, plan paths around the obstacles, and control its motors to ensure that the plan is executed safely. Notice how the final guarantee of collision avoidance depends on each module (the perception, the planning, and the control module) working correctly, *and* interfacing correctly across them.

[3] Lyapunov developed his theories much earlier, around 1892, although the results remained unknown to the western world until around the 1960s

But what does it mean to interface the modules correctly? Beyond just making sure that each module can send and receive appropriate inputs and outputs, we must ensure that the assumptions that each module makes are guaranteed by the previous modules. For example, suppose the planning module *assumes* that obstacles in the world have a position and size that is known exactly. Under said assumption, the planning module guarantees that the planned trajectory is safe. However is this assumption satisfied? If the perception module builds a probabilistic model of the obstacles in the world, the assumption is not satisfied even when the perception module is working perfectly. Similarly, the planning module only guarantees safety if the path was executed perfectly - the controller however may not be able to guarantee this in the presence of disturbances. As such, even when the theory of each module is established, we must ensure that the structure of the assumptions is compatible across the interfaces of each module.

In recent years, and especially with the popularity and capability of the Reinforcement Learning (RL)/Machine Learning (ML) methods, there is a trend towards building end-to-end algorithms for robotic systems. Proponents of this architecture argue that the entire autonomy stack (that is, a single perception-planning-control module) should be simulated and trained to perform their missions while respecting safety constraints. However, the guarantees that such an approach yields are often probabilistic, since the system can only be verified on the scenarios tested in simulation, ultimately a finite set of scenarios [38]. Without tools to extrapolate the systems behavior in unseen scenarios, it is difficult to argue that the system possesses any strong guarantees of safety.

Information Gathering

Finally consider the following example: a Martian rover needs to get to a certain location, but the traversability of the map is unknown. It is unknown whether certain parts of the map are sandy or rocky, and it also unknown what the future weather will be like, and what the available solar resource will be in the future. To ensure that the robot can safely navigate the environment, without violating constraints like battery levels or obstacle avoidance, it must collect information about the environment it is operating in.

At the same time, the robots ability to collect information is affected by the environment and the robots own dynamics. In the sandy parts of the terrain, the robot slips and cannot move as quickly. Furthermore, some pieces of information change temporally (e.g. the solar resource), spatially (e.g. the incline of the ground) or spatiotemporally (e.g., the wind speed at each position and time). To be able to complete the mission successfully, the robot must collect information, but to collect information, the robot must be able to execute a plan

successfully. This interdependence can be challenging for robotic systems to reason about.

The leads us to the following question: given the environment, the robot's dynamics model, and the robot's sensor model, what is the greatest amount of information that can be collected safely? Furthermore, what information is the most valuable to be collected to ensure the robot can operate safely?

1.2 Outline and Contributions

1.2.1 Outline

This thesis is structured primarily into four chapters, where we build our autonomous system architecture in a bottom-up approach.

Ch. 2: We start with the final module, the controller. The controller directly interfaces with the robot, and ultimately determines whether the robot will be safe in the future. In this chapter we pose the problem of safety mathematically, and introduce two controllers that allow for the controller to behave safely especially under input constraints and uncertainty in state estimation.

Ch. 3: In analyzing the controller we will see that to build guarantees of safety, we rely upon the commands sent to the controller being reasonably well-behaved, and satisfying certain assumptions. In this chapter we propose two methods to interface the planning module with the control module. The first method relies on a specific property of the system dynamics (its differential flatness) to prove that the interface between the planner and controller guarantees safety. The second method offers a far more flexible approach to interface planners and controllers, relying instead on the existence of a fail-safe controller.

Ch. 4: Here, we start to address the interface with the perception modules of an autonomy stack. In particular, we demonstrate that one can construct a perception algorithm that can have hard guarantees of correctness (under specific and verifiable assumptions) that are formulated to be compatible with the downstream planning modules described earlier.

Ch. 5: Finally, in this chapter we discuss methods to characterize the maximum information that a robotic system can extract from an operating environment. We introduce the notion of *perceivability* which measures whether a given robotic system (with its specific dynamics and sensor models) can collect information from the environment. We further use these characterizations to construct optimal multiagent controllers to collect information from a spatiotemporal environment.

1.2.2 Top-down vs bottom-up approaches

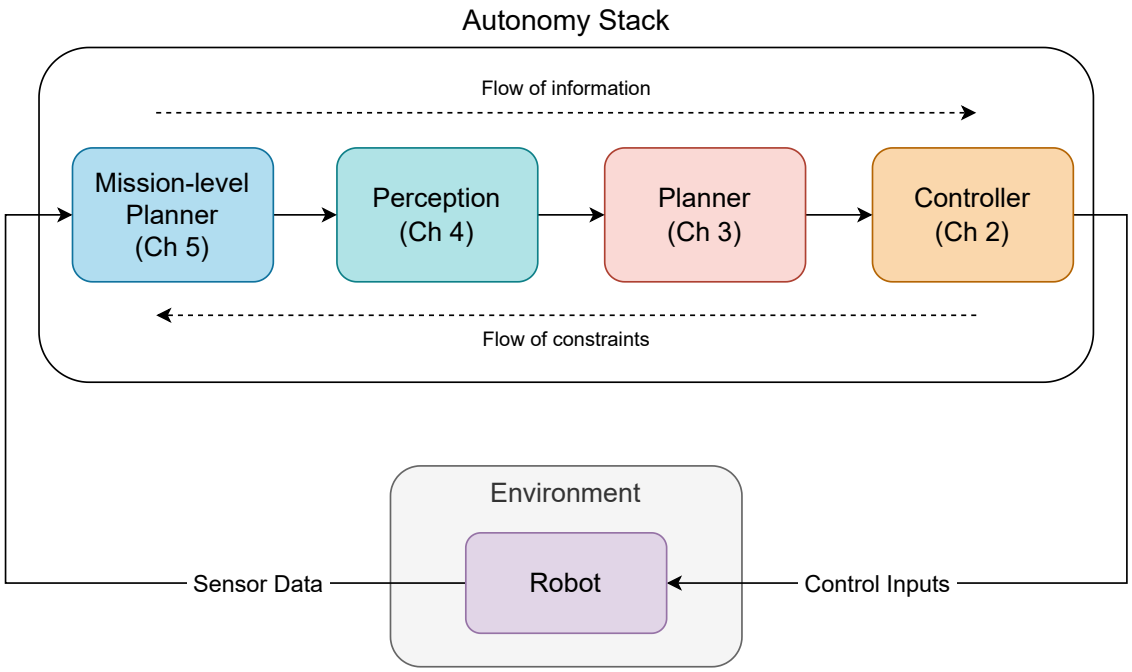

Figure 1.1: A robot's autonomy stack consists of a controller, a planner, a perception module, and (in some cases) an informative path planner. The autonomy stack takes in sensor data, processes it, and computes a control input that is sent to the robot. We explicitly draw the robot as being encased in the environment, since the interaction of the robot with the environment can often determine the behavior of the full system.

We deliberately chose to organize this thesis in this bottom-up sense, instead of the more common top-down approach. Consider the modern autonomy stack, depicted in Figure 1.1. The top-down approach follows from starts by building the mission-level planner to meet mission objectives, and later designs/builds the rest of the perception, planning and control modules. This approach is common and sensible, since it starts from the objectives and eventually designs the full system to meet the objectives. This also corresponds to the flow of information through the autonomy stack, and is often the order in which calculations are performed on a robotic system.

However, for a safety-critical system, it is often important to design the system the other way around, in a bottom-up sense. In a safety-critical autonomy stack, guaranteeing safety is the utmost priority, perhaps even at the expense of not meeting other mission objectives. In such cases, the controller ultimately decides whether safety is maintained. For the controller to be able to do its job, the inputs it receives from the upstream modules must all be

consistent and satisfy the assumptions used by the downstream modules.

Interpreting this backwards flow of constraints is therefore the paradigm shift that is needed to successfully design a safety-critical system: we must design the upstream modules such that they satisfy constraints imposed by downstream modules. Beyond treating this as simply a conceptual shift, in this thesis I argue that in some cases these constraints can and should be imposed explicitly on the upstream modules - this will be especially explicit in Section 3.1, where the low-level tracking controller imposes constraints on the high-level planning module to yield the desired safety guarantees.

The challenge in constructing a safe autonomous system is to balance the top-down flow of mission objectives, with the bottom-up flow of constraints necessary for safety. This requires the safety critical components to be designed with sufficient flexibility that they only constrain the system when necessary to maintain safety. In this thesis, we primarily derive sufficient conditions for safety to be maintained, but we desire the gap between what is necessary for safety and what is sufficient for safety to be small.

In a sense, it can be trivial to design a safety critical system: simply design the controller to never let the robot move. Under mild assumptions (i.e., that the robot starts in a safe place), this controller guarantees safety. However, this controller also prevents the mission from being completed. This idea extends to less trivial scenarios: suppose the safety critical controller of a self-driving vehicle is designed to execute a failsafe stopping maneuver if there are too many cars or pedestrians around for it to perform its computations within a budgeted time - such a controller might be able to maintain safety, but if other cars around it perform a similar algorithm, they may all come to a deadlock scenario, preventing any mission objectives from being satisfied. This example suggests that the safety critical guarantees need to be considered carefully, to ensure that they do not detract from the mission objectives significantly.

1.2.3 Primary Contributions

We now highlight the primary contributions in this thesis, in the order they appear.

1. In Section 2.2 we introduce the notion of an Input-Constrained Control Barrier Function (ICCBF) as developed in [4]. This builds on the CBF-Quadratic Program (QP) controller introduced by Ames in [21], specifically to synthesize controllers to guarantee safety for input-constrained systems.

2. In Section 2.3, we introduce Observer-Robust CBFs, as developed in [5]. This section demonstrates how the CBF-QP controller can be robustified to handle state-estimation

uncertainties, and two approaches are demonstrated depending on the structure of the observer used. This section makes explicit the connection between the controller and the perception module (which often contains the state-estimator).

3. In Section 3.1 we investigate a constructive method to codesign planners and controllers to guarantee safety, as developed in [8]. We investigate deeply how despite different models and assumptions used for the planner and for the controller, the two can be co-designed for safety if the robot posesses a specific property called differential flatness.

4. In Section 3.2 we make the previous approach significantly more flexible and general, and presents the theory developed in [10]. The method is far more flexible since it does not assume differential flatness, but instead assumes the existence of a backup controller, which almost all practical robotic systems will have. The core idea is to filter the output of the path planner before it hits the controller, to ensure that only safe trajectories are sent to the controller.

5. In Section 4.1 we develop a framework to construct certifiably-correct obstacle maps for robotic systems that operate under visual odometry, as presented in [2] (under review at the time of writing). This section identifies that for the guarantees of obstacle avoidance to be valid in a system that only uses visual(-inertial) sensing to localize and map the obstacles in the environment, the state-of-the-art perception modules need to be modified. We present two methods to do this, depending on the mapping methodology used.

6. In Section 5.1 we define two new concepts, clarity and perceivability, as presented in [6].[4] Clarity is an information-theoretic measure based on differential entropy which we demonstrate has desirable properties in the context of informative path planning and dynamic coverage controllers. We use this to develop the notion of perceivability, which measures whether, given an environment to collect data from, a robotic has sufficient sensing capabilities and sufficient actuation capabilities to collect the information. By posing this as an optimal control problem, we can also derive optimal controllers to collect the information.

7. In Section 5.2 we present a multi-agent architecture for a team of robots to explore a domain, collect and assimilate information and determine the optimal control actions, as developed in [7]. It extends the methodologies presented in [6, 126] by using a spatiotemporal Gaussian Process (GP) model of the world, and using the environment

[4]This work was awarded the IEEE Technical Committee on Aerospace Controls (TCAC) Best Paper Award.

model to accurately determine how valuable taking a measurement is on the total amount of information, and therefore determining the optimal control policies for each agent.

The topics presented address some key challenges and bottlenecks in the state-of-the-art methods. These are discussed in detail in their respective chapters, but here we make some broad remarks.

- **Incompatible assumptions:** A significant theme of the methods developed in this thesis is to construct theorems such that the required assumptions are reasonable and satisfied by upstream modules. In state-of-the-art literature it is not uncommon to find results where a module has been designed, but makes assumptions that are unnatural and are not satisfied by the upstream modules. A good example of this is the method proposed in Section 4.1, where instead of improving the perception methods or modifying the mapping methods, we build a modification to the perception algorithm such that the two can be interfaced with guarantees.

- **Safety filtering can be myopic:** Over the last decade, CBFs have become very popular tools to construct formal guarantees of safety. However, as we discuss in Section 2.1, it is (A) challenging to find a CBF for a given system, and (B) even when a CBF is found, it tends to be myopic, i.e., it guarantees safety at the expense of mission objectives. In this thesis, we sought to generalize the idea of set invariance in a way that the safety filtering can be performed at the planning layer instead of at the control layer. This led to the development of the `gatekeeper` framework, discussed in Section 3.2.

- **Fundamental limits of robotic information gathering:** In order for a robot to operate in an environment, it often needs to collect information about the environment, and the state-of-the-art methods on both informative path planning and dynamic coverage implicitly assumed that a robot (or a team of robots) has the ability to collect information from the environment. However from state-of-the-art methods it was not clear whether there was a fundamental limit on the maximum amount of information that can be collected. In Section 5.1, we developed metrics to address this question, and by reformulating information gathering as an optimal control problem were able to derive explicitly the maximum quality of information that can be collected by a robot, and further compute the optimal controller for this purpose. This has also led to significant developments in the theory and algorithms that suitable for dynamic coverage, as discussed in Section 5.2.

1.3 Notation

The following notation is used throughout this thesis, unless otherwise stated.

- *Domains*
  - $\mathbb{N} = \{0, 1, 2, ...\}$ is the set of natural numbers.
  - $\mathbb{Z}$ is the set of integers.
  - $\mathbb{R}, \mathbb{R}_{\geq 0}, \mathbb{R}_{>0}$ denote reals, non-negative reals, and positive reals.
  - $\mathbb{S}^n_+, \mathbb{S}^n_{++}$ denote the set of symmetric positive semi-definite and symmetric positive-definite matrices in $\mathbb{R}^{n \times n}$.
  - $\mathbb{SO}(n)$ is the n-d special orthogonal group.
  - $\mathbb{SE}(n)$ is the n-d special Euclidean group.
  - $\mathbb{B}(r) = \{x \in \mathbb{R}^n : \|x\| \leq r\}$ denotes the closed ball in $\mathbb{R}^n$.
  - $\mathbb{B}(c, r) = \{x \in \mathbb{R}^n : \|x - c\| \leq r\}$ denotes the ball centered at $c \in \mathbb{R}^n$.
  - $\mathbb{L}_\infty(\mathcal{T}, \mathcal{X})$ for $\mathcal{T} \subset \mathbb{R}$, $\mathcal{X} \subset \mathbb{R}^n$ is the set of signals $w : \mathcal{T} \to \mathcal{X}$ such that there exists a $M < \infty$ for which $\|d\|_\infty \leq M$. That is, $\mathbb{L}_\infty(\mathcal{T}, \mathcal{X})$ is the set of bounded signals that map from $\mathcal{T}$ to $\mathcal{X}$.
- *Sets*
  - $\varnothing = \{\ \}$ is the empty set.
  - $A \subset B$ for two sets A, B if $x \in A \implies x \in B$. Note, we do not distinguish between $\subset$ and $\subseteq$ in this thesis.
  - $\text{Int}(\mathcal{C})$ and $\partial\mathcal{C}$ denote the interior and boundary of a set $\mathcal{C}$.
  - $\mathcal{C} \oplus \mathcal{D} = \{c + d \in \mathbb{R}^n : c \in \mathcal{C}, d \in \mathcal{D}\}$ denotes the Minkowski sum between two sets $\mathcal{C}, \mathcal{D} \in \mathbb{R}^n$.
  - $\mathcal{C} \ominus \mathcal{D} = \{c \in \mathcal{C} : c + d \in \mathcal{C}\ \forall d \in \mathcal{D}\}$ denotes the Pontryagin set difference.
- *Vectors*
  - Let $v \in \mathbb{R}^n$.
  - v_i and $[v]_i$ denote the i-th element of $v \in \mathbb{R}^n$.
  - $\|v\|_p = \left(\sum_{i=1}^n |v_i|^p\right)^{1/p}$ refers to the p-norm, for $p \in [1, \infty)$.

- $\|v\|_\infty = \max_{i\in\{1,\dots,n\}} |v_i|$ is the ∞-norm of a vector.
- $\|v\| = \|v\|_2$ is the 2-norm (Euclidean Norm) of $v \in \mathbb{R}^n$.
- $\|v\|_P = \sqrt{v^T P v}$ for $P \in \mathbb{S}^n_{++}$.
- $[v]_\times \in \mathbb{R}^{3\times 3}$ is the skew-symmetric matrix, i.e., the matrix such that $a \times b = [a]_\times b$ for any $a, b \in \mathbb{R}^3$.

- *Matrices*
  - Let $M \in \mathbb{R}^{n\times m}$ be a matrix.
  - $M_{i,j}$ and $[M]_{(i,j)}$ denote the (i,j)-th entry of a matrix $M \in \mathbb{R}^{n\times m}$.
  - $I_n \in \mathbb{R}^{n\times n}$ is the $n \times n$ identity matrix. The subscript is dropped when clear from context.
  - $\|M\|_p = \sup_{x\neq 0} \frac{\|Mx\|_p}{\|x\|_p}$ is the induced p-norm.
  - $\|M\| = \|M\|_2$ is the induced 2-norm of a matrix, unless otherwise specified.
  - $\|M\|_F = \sqrt{\sum_{i=1}^n \sum_{j=1}^m M_{i,j}^2}$ is the Frobenius norm.
  - $\sigma_i(M)$ is the i-th singular value, organized such that $\sigma_1 \geq \sigma_2 \geq \cdots \sigma_k \geq 0$, where $k = \min(n, m)$.
  - $A \otimes B$ denotes the Kronecker product of vectors or matrices A, B.
  - Let $S \in \mathbb{R}^{n\times n}$ be a square matrix.
  - $\lambda_{\min}(S), \lambda_{\max}(S)$ denote the smallest and largest eigenvalues of $S \in \mathbb{R}^{n\times n}$. All eigenvectors are assumed to be unit-norm unless otherwise specified.
  - $|S|$ is the determinant of S.
  - $\mathrm{tr}\,(S)$ is the trace of S.
  - Let $P, Q \in \mathbb{S}^n_+$.
  - $P \geq Q$ if $P - Q \in \mathbb{S}^n_+$
  - $P > Q$ if $P - Q \in \mathbb{S}^n_{++}$.
  - $P^{1/2} \in \mathbb{R}^{n\times n}$ denotes the (unique) matrix square-root of $P \in \mathbb{S}^n_{++}$, i.e., the matrix such that $P^{1/2}P^{1/2} = P$.

- *Functions*
  - $\|w\|_\infty = \sup_{t\in\mathcal{T}} \|w(t)\|$ for a signal $w : \mathcal{T} \to \mathbb{R}^n$ where $\mathcal{T} \subset \mathbb{R}$.

– $L_f h(x)$ denotes the Lie derivatives of a scalar function $h : \mathbb{R}^n \to \mathbb{R}$, along a vector field $f : \mathbb{R}^n \to \mathbb{R}^n$, $L_f h(x) = \frac{\partial h}{\partial x}(x) f(x)$. If vector fields has an additional dependency, e.g., $f : \mathbb{R}^n \times \mathbb{R}^p \to \mathbb{R}^n$, the notation $L_f h(x, y) = \frac{\partial h}{\partial x}(x) f(x, y)$ is used.

Where possible, the following conventions are adopted. Calligraphic letters, e.g. $\mathcal{S}$,$\mathcal{U}$ represent sets. Lower case Greek letters represent scalars or scalar-valued functions. Latin characters represent vectors or vector-valued functions. Lowercase letters are used for vectors, while uppercase are used for matrices. Time is denoted by t or τ, and T or Δt will denote a duration of time. These conventions are not strictly followed.

We also use the following definitions [20, 90]:

Definition 1.1 (Class $\mathcal{K}$)**.** *A function $\alpha : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ is said to be of class $\mathcal{K}$, ($\alpha \in \mathcal{K}$), if it is continuous, $\alpha(0) = 0$, and strictly-increasing.*

Definition 1.2 (Class $\mathcal{K}_\infty$)**.** *A function $\alpha : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ is said to be of class $\mathcal{K}_\infty$, ($\alpha \in \mathcal{K}_\infty$), if $\alpha \in \mathcal{K}$ and $\lim_{r \to \infty} \alpha(r) = \infty$.*

Definition 1.3 (Class $\mathcal{K}_e$)**.** *, A function $\alpha : \mathbb{R} \to \mathbb{R}$ is said to be of class $\mathcal{K}_e$, ($\alpha \in \mathcal{K}_e$), if it is continuous, $\alpha(0) = 0$, and α is strictly increasing.*

Definition 1.4 (Class $\mathcal{L}$)**.** *A function $\sigma : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ is said to be of class $\mathcal{L}$, ($\sigma \in \mathcal{L}$), if it is continuous, strictly decreasing,*[5] *and $\lim_{s \to \infty} \sigma(s) = 0$.*

Definition 1.5 (Class $\mathcal{KL}$)**.** *A function $\beta : \mathbb{R}_{\geq 0} \times \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ is said to be of class $\mathcal{KL}$, ($\beta \in \mathcal{KL}$), if it is class $\mathcal{K}$ in its first argument, and class $\mathcal{L}$ in its second.*

[5] Note some authors prefer non-increasing.

CHAPTER 2

Safety-Critical Control

2.1 Safe Control as a Set Invariance Problem

In this section, we establish definitions for "safety," and show how the synthesis of safe control policies is specified mathematically. Following this, we highlight the primary tools and foundational results used throughout the thesis.

2.1.1 Safety and Forward Invariance

Consider a dynamical system of the form

$$\dot{x} = F(t, x, u) \tag{2.1}$$

where $t \in \mathbb{R}$ is time, $x \in \mathcal{X} \subset \mathbb{R}^n$ is the system state, and $u \in \mathcal{U} \subset \mathbb{R}^m$ is the control input.

Under a feedback policy $\pi : \mathcal{T} \times \mathcal{X} \to \mathcal{U}$, the closed-loop system is

$$\dot{x} = F_{\text{cl}}(t, x) = F(t, x, \pi(t, x)). \tag{2.2}$$

For a given initial condition $x(t_0) = x_0 \in \mathcal{X}$, if F_{cl} is piecewise-continuous in t and locally Lipschitz in x, there exists an interval $[t_0, t_1)$ such that a unique solution exists to the Ordinary Differential Equation (ODE). If we further assume F_{cl} is Lipschitz in x for all $x \in \mathcal{X}$, then a unique solution exists for all $t \geq t_0$. [92, Ch. 3]. For the remainder of this section, assume unique solutions exist for all $t \geq t_0$.

Safety can be formalized through the notion of a 'safe set,' often denoted by $\mathcal{S} \subset \mathcal{X}$.[1] The idea is that given a set of specifications (or equivalently, constraints) on what it means for a robotic system to be operating safely, one can define a set of states, $\mathcal{S}$, such that if the robot state $x \in \mathcal{S}$, we say that the robot is safe, at least instantaneously.

[1] All of the following definitions can be extended to the case where $\mathcal{S}$ is not a static set, but one that changes with time. This will be discussed further in Section 3.2.

Definition 2.1 (Instantaneously Safe)**.** *A state $x \in \mathcal{X}$ is **instantaneously safe** if $x \in \mathcal{S}$.*

This abstraction allows the safe set $\mathcal{S}$ to be defined in various ways depending on the intended application. For instance, in an obstacle avoidance problem, the set $\mathcal{S}$ is the set of states such that the robot is not in collision with an obstacle. In a multi-agent path planning problem, the set $\mathcal{S}$ is the set of states where the inter-agent distance is above some threshold. In a problem satellite docking problem, the set $\mathcal{S}$ could be the set of states where the chaser satellite lies within a docking cone of the target satellite. The set $\mathcal{S}$ can also be defined for multiple constraints at the same time. For instance, in a self-driving scenario $\mathcal{S}$ could be the set of states that do not collide with nearby vehicles, *and* do not exceed the speed limit, *and* stay within the lane. Multiple constraints can significantly complicate the analysis, but conceptually, the intersection of the allowable states for each constraint can be thought of as the safe set $\mathcal{S}$.

Although a state may be instantaneously safe, this does not mean that the system will be safe at future time. For instance, consider the obstacle avoidance scenario: if the robot is near (but not yet in collision with) an obstacle, and is moving with high velocity towards the obstacle, it may not be possible for the robot to avoid collision. In this way, although the robot is instantaneously safe, it may not be possible to ensure the robot remains safe in the future. Therefore, we extend the notions of safety to consider a trajectory of the system:

Definition 2.2 (Safe Trajectory)**.** *A trajectory $x : \mathcal{T} \to \mathcal{X}$, defined over $\mathcal{T} \subset \mathbb{R}$, is a **safe trajectory** if*

$$x(t) \in \mathcal{S} \ \forall t \in \mathcal{T}. \tag{2.3}$$

The goal then is to design a control policy $u = \pi(t, x)$ such that closed-loop trajectories from the possible initial conditions are all safe trajectories. Our primary tool for verifying if a control policy is safe is forward invariance:

Definition 2.3 (Controlled Forward Invariance)**.** *A controller $\pi : [t_0, \infty) \times \mathcal{X} \to \mathcal{U}$ **renders a set $\mathcal{C} \subset \mathcal{X}$ forward invariant** if for all $t_0 \in \mathbb{R}$ and all initial conditions $x_0 \in \mathcal{C}$ the closed-loop system*

$$\begin{cases} \dot{x} = F_{\text{cl}}(t, x) = F(t, x, \pi(t, x)), \\ x(t_0) = x_0 \end{cases} \tag{2.4}$$

satisfies

$$x(t_0) \in \mathcal{C} \implies x(t) \in \mathcal{C} \tag{2.5}$$

for all $t \geq t_0$. The set $\mathcal{C}$ is a ***controlled-invariant set****.*

This leads us to the notion of a *safe controller*:

Definition 2.4 (Safe Controller)**.** *A controller $\pi : \mathcal{T} \times \mathcal{X} \to \mathcal{U}$ is a* ***safe controller*** *if it renders a set $\mathcal{C}$ forward invariant, and $\mathcal{C} \subset \mathcal{S}$.*

Notice that we introduced a set $\mathcal{C} \subset \mathcal{X}$ in the above definition. This is precisely because we must distinguish between $\mathcal{S}$, the set of instantaneously safe states, and $\mathcal{C}$, the set of states that can be rendered forward invariant by a controller π. Note, the set $\mathcal{C}$ need not be unique: for any given $\mathcal{S}$ there can be multiple $\mathcal{C}$ sets, each associated with its own controller π.

Before we continue, we make a small remark about the existence of unique solutions for all time. In the above, we have assumed that a unique solution exists for all $t \geq t_0$. It is important to ensure that the chosen controller does not violate this assumption. Consider the following example [32, Sec 4.1]:

Example 2.1. Consider a scalar system $\dot{x} = u$, and let $\mathcal{S} = \{x \in \mathbb{R} : x \geq 0\}$. Then, it is seemingly obvious that any controller $u = \pi(x) \geq 0$ guarantees that $x(t_0) \geq 0 \implies x(t) \geq 0$ for all $t \geq 0$. However consider a controller $\pi(x) = 1 + x^2$. In this case, the solution of the closed-loop system from an initial condition $x(t_0) = 0$ is $x(t) = \tan(t)$, which although positive, also has finite escape time. Since it doesn't satisfy $x(t) \in \mathcal{S}$ *for all* $t \geq t_0$, π is not considered a safe controller.

2.1.2 Nagumo's Theorem

Nagumo's theorem [125] provides the necessary and sufficient conditions to establish whether a closed-loop system remains within a set. Although the original is in German, Blanchini [32] provides an excellent and modern presentation of the ideas.

Let the distance from a set be defined as follows:

Definition 2.5 (Distance from a set, [32, Def 4.5])**.** *Given a set $\mathcal{S} \subset \mathbb{R}^n$, and a point $y \in \mathbb{R}^n$, the* ***distance to $\mathcal{S}$*** *is*

$$\operatorname{dist}(y, \mathcal{S}) = \inf_{x \in \mathcal{S}} \|y - x\| . \tag{2.6}$$

We can now define the tangent cone of a set:

Definition 2.6 (Bouligand Tangent Cone, [32, Def. 4.6])**.** *Given a closed set $\mathcal{S} \subset \mathbb{R}^n$, the* ***Bouligand Tangent Cone*** *of $\mathcal{S}$ at $x \in \mathbb{R}^n$ is*

$$\mathcal{T}_{\mathcal{S}}(x) = \left\{ v \in \mathbb{R}^n : \liminf_{\tau \to 0} \frac{\operatorname{dist}(x + \tau v, \mathcal{S})}{\tau} = 0 \right\} . \tag{2.7}$$

Notice that for any $x \in \operatorname{Int} \mathcal{S}$, $\mathcal{T}_{\mathcal{S}}(x) = \mathbb{R}^n$ and for any $x \notin \mathcal{S}$, $\mathcal{T}_{\mathcal{S}}(x) = \varnothing$. Therefore, $\mathcal{T}_{\mathcal{S}}(x)$ is non-trivial only when $x \in \partial\mathcal{S}$, that is, x is on the boundary of $\mathcal{S}$. In such cases, the tangent cone defines the directions we can perturb x but still remain within $\mathcal{S}$.

Finally, we present Nagumo's theorem:[2]

Theorem 2.1 (Nagumo's theorem, [32, Cor. 4.8])**.** *Consider a system $\dot{x} = F(x)$, and assume that for each initial condition $x(0) \in \mathcal{O}$ in an open set $\mathcal{O} \subset \mathbb{R}^n$, it admits a unique solution defined for all $t \geq 0$. Let $\mathcal{S} \subset \mathcal{O}$ be a closed set. Then $\mathcal{S}$ is positively invariant if and only if*

$$f(x) \in \mathcal{T}_{\mathcal{S}}(x) \quad \forall x \in \mathcal{S}. \tag{2.8}$$

Nagumo's theorem provides necessary and sufficient conditions on the closed-loop dynamics to ensure that the system remains within a set $\mathcal{S}$. The task of designing safe controllers is therefore seemingly straightforward: design a controller such that at the boundary of the safe set, the closed-loop dynamics lie within the tangent cone of $\mathcal{S}$. This turns out to be not so trivial (and also not sufficient to guarantee safety). The remainder of the thesis is devoted to designing such a controller, and extending the setting of Nagumo's theorem to be applicable for real-world robotic systems.

2.1.3 Control Barrier Functions (CBFs)

Over the past decade the notion of a CBF has gained traction since it provides an intuitive and powerful method to synthesize safe controllers for nonlinear systems. The notion of a CBF originates with the ideas of a Barrier Certificate, as presented in [132], and of a Barrier Function [170]. The original papers on CBFs introduced both Reciprocal Control Barrier Functions and Zeroing Control Barrier Functions. In recent years, the Zeroing CBFs have become popular, and most extensions are now based on these CBFs. We review the zeroing CBF formulation of safety next. Compared to (2.1) we will assume some additional structure.

Consider a (time-invariant) control-affine dynamical system

$$\dot{x} = f(x) + g(x)u \tag{2.9}$$

where, as before, $x \in \mathcal{X} \subset \mathbb{R}^n$ is the state and $u \in \mathcal{U} \subset \mathbb{R}^m$ is the control input. Suppose $f : \mathcal{X} \to \mathbb{R}^n$, $g : \mathcal{X} \to \mathbb{R}^{n \times m}$ are both locally Lipschitz.

[2] Although presented for autonomous and uncontrolled systems, this can be extended to the non-autonomous controlled setting in a straightforward way [31, Sec. 4.2].

Furthermore, suppose a set $\mathcal{C}$ can be defined as the zero-superlevel set of a function $h : \mathcal{X} \to \mathbb{R}$:

$$\mathcal{C} = \{x \in \mathcal{X} : h(x) \geq 0\} \tag{2.10a}$$

$$\partial\mathcal{C} = \{x \in \mathcal{X} : h(x) = 0\} \tag{2.10b}$$

$$\operatorname{Int}\mathcal{C} = \{x \in \mathcal{X} : h(x) > 0\} \tag{2.10c}$$

One should think of $\mathcal{C}$ as a subset of the safe set $\mathcal{S}$.

Definition 2.7 (Control Barrier Function, [21, Def. 5])**.** *Given a set $\mathcal{C}$ defined as in* (2.10) *by a continuously differentiable function $h : \mathcal{X} \to \mathbb{R}$, the function h is a* ***(zeroing) Control Barrier Function (CBF)*** *defined on a set $\mathcal{D}$ with $\mathcal{C} \subset \mathcal{D} \subset \mathcal{X}$ for the system* (2.9)*, if there exists a class $\mathcal{K}_e$ function α such that*

$$\sup_{u \in \mathcal{U}} \left(L_f h(x) + L_g h(x) u\right) \geq -\alpha(h(x)) \quad \forall x \in \mathcal{D}, \tag{2.11}$$

and $\partial h / \partial x \neq 0$ for any $x \in \partial\mathcal{C}$.

Recall class $\mathcal{K}_e$ was defined in Definition 1.3, and $L_f h : \mathcal{X} \to \mathbb{R}$, $L_g h : \mathcal{X} \to \mathbb{R}^{1\times m}$ are defined as

$$L_f h(x) = \frac{\partial h}{\partial x}(x) f(x) \quad L_g h(x) = \frac{\partial h}{\partial x}(x) g(x).$$

Given a CBF h, we can define a set-valued map $K_{\text{cbf}} : \mathcal{D} \rightrightarrows \mathcal{U}$ as

$$K_{\text{cbf}}(x) = \{u \in \mathcal{U} : L_f h(x) + L_g h(x) u \geq -\alpha(h(x))\} \tag{2.12}$$

for any $x \in \mathcal{D}$.

Ames [21] established a sufficient condition for a controller to be a safe controller:

Theorem 2.2 ([21, Cor. 2])**.** *Given a set $\mathcal{C}$ as defined in* (2.10)*, if h is a CBF on $\mathcal{D}$ for the system* (2.9)*, then any Lipschitz continuous controller $\pi : \mathcal{D} \to \mathcal{U}$ such that $\pi(x) \in K_{\text{cbf}}(x)$ for all $x \in \mathcal{D}$ will render the set $\mathcal{C}$ forward invariant.*

Naturally, if $\mathcal{C} \subset \mathcal{S}$, then π is a safe controller.

The benefit of this construction is that the requirement that $\pi(x) \in K_{\text{cbf}}(x)$ is an affine constraint on the possible values of u, *regardless of whether* f, g, h are nonlinear functions. This makes it appealing for nonlinear systems, since we can construct an optimization-based controller of the following form

Definition 2.8 (CBF-QP Safety Filter)**.** *Let $\pi_d : \mathcal{X} \to \mathcal{U}$ be a desired (but possibly unsafe) controller for the system* (2.9)*. Given a set $\mathcal{C} \subset \mathcal{S}$ as defined in* (2.10)*, if h is a CBF on $\mathcal{D}$ for the system* (2.9)*, the controller $\pi : \mathcal{D} \to \mathcal{U}$ defined by*

$$\pi(x) = \underset{u \in \mathcal{U}}{\operatorname{argmin}} \quad \|u - \pi_d(x)\|^2 \tag{2.13a}$$

$$\text{subject to} \quad L_f h(x) + L_g h(x) u \geq -\alpha(h(x)) \tag{2.13b}$$

*is a **CBF-QP safety filter**.*

Using Theorem 2.2, we can conclude the following (a similar result is presented in [178])

Corollary 2.3 (CBF-QP)**.** *Suppose π is a CBF-QP safety filter as in Definition 2.8. Further, assume $\mathcal{U} = \mathbb{R}^m$ is unbounded, and $\pi_d : \mathcal{X} \to \mathcal{U}$ is Lipschitz continuous. Then $\pi : \mathcal{D} \to \mathcal{U}$ is also Lipschitz continuous, and π is a safe controller for the system* (2.9) *on the set $\mathcal{S}$.*

The implementation of the the safety filter is remarkably simple: (2.13) is a QP parameterized by x that can be solved very efficiently using libraries like [157], or analytically [178].

Remark 2.1. *The optimization-based controller* (2.13) *has the analytic solution*

$$\pi(x) = \begin{cases} \pi_d(x) & \text{if } \omega(x) \geq 0, \\ \pi_d(x) - \frac{\omega(x)}{\|L_g h(x)\|^2} L_g h(x)^T & \text{else.} \end{cases} \tag{2.14}$$

where $\omega(x) = L_f h(x) + L_g h(x) \pi_d(x) + \alpha(h(x))$. Equivalently,

$$\pi(x) = \pi_d(x) - \frac{\min(0, \omega(x))}{\|L_g h(x)\|^2} L_g h(x)^T \tag{2.15}$$

The fundamental challenge with using the CBF-QP is that finding a CBF for a given problem is not trivial. Although the community has developed many extensions of CBFs to address real-world limitations, and even despite the presence of numerical methods to find CBFs, in practice it remains a challenge to construct and validate a CBF for a given application. See [20, 69] for a review of recent developments in this area.

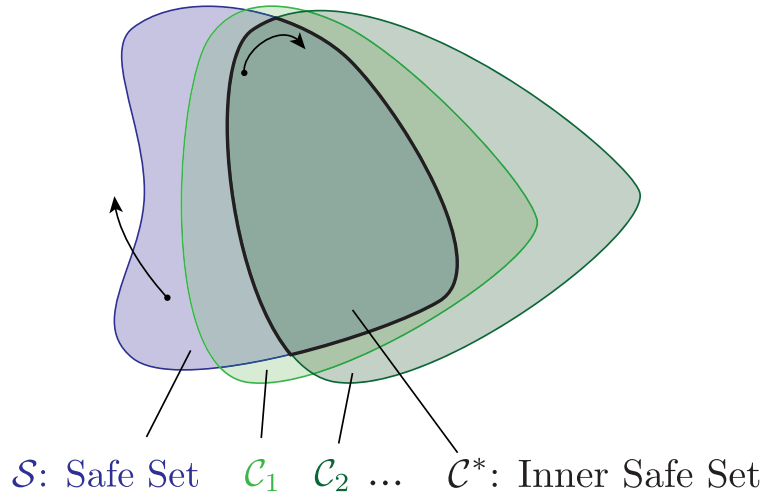

Figure 2.1: Visual representation of ICCBF method. The safe set $\mathcal{S}$ and two intermediate sets $\mathcal{C}_1$ and $\mathcal{C}_2$ are drawn. The final inner safe set $\mathcal{C}^*$ is the intersection of each of these sets, and can be rendered forward invariant.

2.2 Input-Constrained Control Barrier Function (ICCBF)

Prior work on CBFs [19, 20, 67, 71] has largely focused on systems where a sufficiently large control authority is available to ensure forward invariance of the safe set. However in the presence of input constraints, only a subset of the safe set may be rendered forward invariant, which we term the inner safe set. A few methods have been proposed to find the inner safe set. These include reachability analysis by solving a Hamilton-Jacobi equations [71, 119] and Sum of Squares (SOS), which employ the positivstellensatz theorem to provide a certificate of safety [33, 177]. Both methods scale poorly with the dimension of the state-space. Some methods have also been proposed for specific classes of systems, e.g. Euler-Lagrange systems [53] or mechanical systems in a manifold [173].

In this section, we introduce the notion of an ICCBF. We show that an ICCBF guarantees that an input constrained controller can render the super-level set of the ICCBF forward invariant. Furthermore, we show that ICCBFs represent a generalization of Higher Order Control Barrier Functions (HOCBFs) [176], enabling synthesis of input-constrained controllers for safety functions of non-uniform relative degree. Finally, the method is applied to an adaptive cruise control problem [18], and a spacecraft rendezvous problem, demonstrating that ICCBFs define a safe controller that respects input constraints.

2.2.1 Problem Formulation and Preliminaries

Problem Setup

Consider a nonlinear, control-affine dynamical system, with state $x \in \mathcal{X} \subset \mathbb{R}^n$ and control input $u \in \mathcal{U} \subset \mathbb{R}^m$

$$\dot{x} = f(x) + g(x)u, \tag{2.16}$$

where $f : \mathcal{X} \to \mathbb{R}^n$, $g : \mathcal{X} \to \mathbb{R}^{n \times m}$ are sufficiently smooth, as will be discussed in 2.2.2. We assume these functions are known, and the system state is measured exactly. Under a Lipschitz continuous feedback law $u = \pi(x)$, the closed-loop system is

$$\dot{x} = f(x) + g(x)\pi(x). \tag{2.17}$$

We define a state x as *safe*, if it lies in a set $\mathcal{S}$, the 0-superlevel set of a continuously differentiable function $h : \mathcal{X} \to \mathbb{R}$:

$$\mathcal{S} \triangleq \{x \in \mathcal{X} : h(x) \geq 0\} \tag{2.18}$$

$$\partial\mathcal{S} \triangleq \{x \in \mathcal{X} : h(x) = 0\} \tag{2.19}$$

$$\text{Int}(\mathcal{S}) \triangleq \{x \in \mathcal{X} : h(x) > 0\} \tag{2.20}$$

The set $\mathcal{S}$ is referred to as the *safe* set. We assume this set is closed, non-empty and simply connected. Recall the definition of forward invariance as in Definition 2.3 adapted to this setting:

Definition 2.9. *A set $\mathcal{S}$ is rendered **forward invariant** by a feedback controller $\pi : \mathcal{S} \to \mathcal{U}$, if for the closed-loop system (2.17), $x(0) \in \mathcal{S}$ implies $x(t) \in \mathcal{S}$ for all $t \geq 0$.*

Due to input constraints however, there may not exist a controller which renders the safe set forward invariant (Example 2.2). We propose the definition of an *inner safe set*.

Definition 2.10. *A non-empty closed set $\mathcal{C}^*$ is an **inner safe set** of the safe set $\mathcal{S}$ for the dynamical system (2.16), if $\mathcal{C}^* \subseteq \mathcal{S}$ and there exists a feedback controller $\pi : \mathcal{C}^* \to \mathcal{U}$ such that $\mathcal{C}^*$ is rendered forward invariant by π.*

Example 2.2. Consider the following scalar dynamical system with input and safety constraints:

$$\dot{x} = x + u, \quad \mathcal{U} = [-1, 1], \quad \mathcal{S} = \{x \in \mathbb{R} : x \leq 2\}$$

i.e. $h(x) = 2 - x$. Now consider the boundary state $x = 2 \in \mathcal{S}$. Since

$$\dot{h} = -2 - u \implies \dot{h} \leq -1 \; \forall u \in \mathcal{U},$$

i.e., all closed-loop trajectories starting at $x(0) = 2$ leave the safe set. Thus $\mathcal{S}$ cannot be rendered forward invariant. The set $\mathcal{C}^* = \{x : x \leq 1\}$ is an inner safe set. $\triangle$

Now, we can state the main objective:

Problem 2.1. *Given the system (2.16), find a closed set $\mathcal{C}^* \subseteq \mathcal{S}$ and a feedback controller $\pi : \mathcal{C}^* \to \mathcal{U}$, such that for any $x(0) \in \mathcal{C}^*$, the closed-loop trajectories of (2.17) satisfy $x(t) \in \mathcal{C}^*$ for all $t \geq 0$.*

In words, the objective is to find a subset of the safe set, and a corresponding feedback controller that renders the subset forward invariant.

Set Invariance

Nagumo's theorem provides a necessary and sufficient condition for the forward invariance of a set $\mathcal{S}$. In this setting, Nagumo's theorem simplifies to:

Lemma 2.4. *Consider the system (2.16). Let the set $\mathcal{S}$ be defined by a continuously differentiable function $h : \mathcal{X} \to \mathbb{R}$, as per (2.18-2.20). Consider a Lipschitz continuous feedback controller $\pi : \mathcal{S} \to \mathcal{U}$, such that for any initial condition $x(0) \in \mathcal{S}$, the closed-loop system (2.17) admits a globally unique solution. Then set $\mathcal{S}$ is forward invariant if and only if*

$$L_f h(x) + L_g h(x)\pi(x) \geq 0 \quad \forall x \in \partial\mathcal{S}. \tag{2.21}$$

In [19, 20], a stronger notion of the control barrier function is introduced:

Definition 2.11 (Control Barrier Function [20])**.** *Let $S \subset \mathcal{X} \subset \mathbb{R}^n$ be the superlevel set of a continuously differentiable function $h : \mathcal{X} \to \mathbb{R}$. h is a **CBF** if there exists a class $\mathcal{K}_{e,\infty}$ function α such that for the control system* (2.16)*:*

$$\sup_{u \in \mathcal{U}} [L_f h(x) + L_g h(x) u] \geq -\alpha(h(x)) \tag{2.22}$$

for all $x \in \mathcal{X}$.

Lemma 2.5 ([20], Theorem 2)**.** *Let $\mathcal{S} \subset \mathbb{R}^n$ be a set defined as the superlevel set of a continuously differentiable function $h : \mathcal{X} \subset \mathbb{R}^n \to \mathbb{R}$. If h is a CBF on $\mathcal{X}$, and $dh/dx(x) \neq 0$*

for all $x \in \partial\mathcal{C}$, then any Lipschitz continuous controller $\pi(x) \in K_{\text{CBF}}$, where

$$K_{\text{CBF}}(x) = \{u \in \mathcal{U} : L_f h(x) + L_g h(x) u + \alpha(h(x)) \geq 0\}, \tag{2.23}$$

for the control system (2.17), renders the set $\mathcal{S}$ safe.

In this section, we focus on cases where $h(x)$ defining the safe set $\mathcal{S}$ is *not* a valid control barrier function.

2.2.2 Main Result

In this section we define Input Constrained Control Barrier Functions (ICCBFs). To aid the reader, first the method is explained conceptually, and formal definitions are presented second.

Motivation

Suppose the safe set $\mathcal{S}$ associated with h cannot be rendered forward invariant by any feedback controller $\pi(x)$, since there exist some states where it would require $u \notin \mathcal{U}$ to render safe. We wish to remove these states from $\mathcal{S}$. We define a function $b_1 : \mathcal{X} \to \mathbb{R}$ and a set $\mathcal{C}_1$ (visualized in Figure 2.1) as follows

$$b_1(x) = \inf_{u \in \mathcal{U}} [L_f h(x) + L_g h(x) u + \alpha_0(h(x))] \tag{2.24}$$

$$\mathcal{C}_1 = \{x \in \mathcal{X} : b_1(x) \geq 0\} \tag{2.25}$$

where α_0 is some user specified class $\mathcal{K}$ function. Since an infimum over $\mathcal{U}$ is taken, b_1 only depends on x, and not u.

The set $\mathcal{C}_1$ has a useful property: Suppose there exists a point $x \in \partial\mathcal{S}$ and $x \in \mathcal{C}_1$, i.e., $h(x) = 0$ and $b_1(x) \geq 0$. Then, from (2.24),

$$x \in \partial\mathcal{S} \cap \mathcal{C}_1 \implies \inf_{u \in \mathcal{U}} [L_f h(x) + L_g h(x) u] \geq 0 \tag{2.26}$$

$$\implies L_f h(x) + L_g h(x) u \geq 0, \ \forall u \in \mathcal{U}. \tag{2.27}$$

Thus the closed-loop trajectory cannot leave $\mathcal{S}$ through x. Notice that if there exists a Lipschitz continuous controller π which renders $\mathcal{C}_1$ forward invariant, it is immediate $\mathcal{S} \cap \mathcal{C}_1$ is also forward invariant: any $x(t)$ that reaches the boundary $\partial\mathcal{S}$ must lie in $\mathcal{C}_1$ (by assumption on π), and thus by (2.27), $x(t)$ also cannot leave $\mathcal{S}$.

The problem now is to find the controller that renders $\mathcal{C}_1$ forward invariant. If this cannot be done, the steps can be repeated: define $b_2(x) = \inf_{u\in\mathcal{U}}[\dot{b}_1(x,u) + \alpha_1(b_1(x))]$ and $\mathcal{C}_2 = \{x \in \mathcal{X} : b_2(x) \geq 0\}$. Now any controller that renders $\mathcal{C}_2$ forward invariant also renders $\mathcal{C}_1 \cap \mathcal{C}_2$ forward invariant, and therefore the set $\mathcal{C}^* = \mathcal{S} \cap \mathcal{C}_1 \cap \mathcal{C}_2$ is also forward invariant *by the same controller*. This idea is formalized in the next subsection.

ICCBFs

Consider the dynamical system (2.16) with bounded control inputs $u \in \mathcal{U}$ and a safe set $\mathcal{S}$ defined by a function $h : \mathcal{X} \to \mathbb{R}$, as per (2.18-2.20). We define the following sequence of functions:

$$b_0(x) = h(x) \tag{2.28a}$$
$$b_1(x) = \inf_{u\in\mathcal{U}}[L_f b_0(x) + L_g b_0(x)u + \alpha_0(b_0(x))] \tag{2.28b}$$
$$\vdots$$
$$b_N(x) = \inf_{u\in\mathcal{U}}[L_f b_{N-1}(x) + L_g b_{N-1}(x)u + \alpha_{N-1}(b_{N-1}(x))], \tag{2.28c}$$

where each α_i is a class $\mathcal{K}$ function, and N is a positive integer. We assume the functions f, g, h are sufficiently smooth such that b_N and its derivative are defined. The time derivative $\dot{b}_i = L_f b_i(x) + L_g b_i(x)u$ is still affine in u. Next, we define a family of sets,

$$\mathcal{C}_0 = \{x \in \mathcal{X} : b_0(x) \geq 0\} = \mathcal{S} \tag{2.29a}$$
$$\mathcal{C}_1 = \{x \in \mathcal{X} : b_1(x) \geq 0\} \tag{2.29b}$$
$$\vdots$$
$$\mathcal{C}_N = \{x \in \mathcal{X} : b_N(x) \geq 0\}. \tag{2.29c}$$

The intersection of these sets is $\mathcal{C}^*$, assumed closed, non-empty and without isolated points:

$$\mathcal{C}^* = \mathcal{C}_0 \cap \mathcal{C}_1 \cap ... \cap \mathcal{C}_N. \tag{2.30}$$

Definition 2.12. *For the dynamical system* (2.17) *with safe set* $\mathcal{S}$ *and continuously differentiable class* $\mathcal{K}$ *functions* $\alpha_0, ..., \alpha_{N-1}$*, if there exists a class* $\mathcal{K}$ *function* α_N *such that*

$$\sup_{u\in\mathcal{U}}[L_f b_N(x) + L_g b_N(x)u + \alpha_N(b_N(x))] \geq 0 \ \forall x \in \mathcal{C}^*, \tag{2.31}$$

then b_N *is an* ***ICCBF****.*

Note, this does not require b_N to be a CBF on $\mathcal{C}_N$. The definition only requires condition (2.31) to hold for $x \in \mathcal{C}^*$, a subset of $\mathcal{C}_N$.

Theorem 2.6 (Main Result)**.** *Given the input constrained dynamical system (2.16), if b_N is an ICCBF, then any Lipschitz continuous controller $\pi : \mathcal{C}^* \to \mathcal{U}$ such that $\pi(x) \in K_{\text{ICCBF}}(x)$, where*

$$K_{\text{ICCBF}}(x) = \{u \in \mathcal{U} : L_f b_N(x) + L_g b_N(x) u \geq -\alpha_N(b_N(x))\} \tag{2.32}$$

renders the set $\mathcal{C}^* \subseteq \mathcal{S}$ (2.30) forward invariant.

Proof. Since u is a Lipschitz continuous controller, the closed-loop system (2.17) is also Lipschitz continuous. To show forward invariance of $\mathcal{C}^*$, we use Nagumo's theorem on the closed-loop system. In particular, we show that

$$\begin{aligned} &x \in \mathcal{C}^*, \pi(x) \in K_{\text{ICCBF}}(x) \text{ and } b_i(x) = 0 \\ &\quad \implies \frac{db_i}{dx}[f(x) + g(x)\pi(x)] \geq 0, \end{aligned} \tag{2.33}$$

We show (2.33) holds for each $i \in \mathcal{I}(x) = \{i : b_i(x) = 0\}$:

Cases $i \in \{0, ..., N-1\}$: Consider any $x \in \mathcal{C}^* \cap \partial\mathcal{C}_i$. Since $\mathcal{C}^* \subseteq \mathcal{C}_{i+1}$, $x \in \partial\mathcal{C}_i \cap \mathcal{C}_{i+1}$. By (2.28, 2.29),

$$\begin{aligned} &\inf_{u \in \mathcal{U}} [L_f b_i(x) + L_g b_i(x) u] \geq 0 \\ &\quad \therefore L_f b_i(x) + L_g b_i(x) u \geq 0, \quad \forall u \in \mathcal{U} \end{aligned} \tag{2.34}$$

and since $\pi(x) \in K_{\text{ICCBF}}(x) \subseteq \mathcal{U}$, (2.33) is satisfied.

Case $i = N$: Consider $x \in \mathcal{C}^* \cap \partial\mathcal{C}_N$. Since b_N is an ICCBF and $b_N(x) = 0$, by (2.29c, 2.32),

$$L_f b_N(x) + L_g b_N(x)\pi(x) \geq 0 \ \forall \pi(x) \in K_{\text{ICCBF}}(x), \tag{2.35}$$

thus satisfying (2.33).

In conclusion, we have shown that condition (2.33) is satisfied for all $i \in \mathcal{I}(x)$, and therefore the conditions of Nagumo's theorem are satisfied, completing the proof. □

Remark 2.2. *The practical value of this construction is that for a given system, a set $\mathcal{S}$ of safe states of practical importance can be specified, which may not be rendered forward invariant under the given system dynamics. By using ICCBFs, we remove some states from*

the set $\mathcal{S}$, and construct an inner set for which we can find a controller that renders it forward invariant.

Remark 2.3. *A quadratic program based feedback controller can be used for polytopic input constraints, $\mathcal{U} = \{u : Pu \leq q\}$:*

$$\begin{aligned} \pi(x) = \underset{u \in \mathbb{R}^m}{\text{argmin}} \quad & u^T u \\ \text{subject to} \quad & L_f b_N(x) + L_g b_N(x) u \geq -\alpha_N(b_N(x)) \\ & Pu \leq q, \end{aligned}$$

provided suitable regularity conditions hold, for instance $L_g b_N(x)$ is linearly independent of the rows of P [75, 122]. Note, this QP is always guaranteed to be feasible.

We would like to note a useful special case, the *simple ICCBF*:

Definition 2.13. *In the above construction, if $\mathcal{C}^*$ is a strict subset of $\mathcal{C}_N$, i.e., $\mathcal{C}^* \subset \mathcal{C}_N$, then b_N is a **simple ICCBF**.*

Theorem 2.7. *For the dynamical system (2.16), if b_N is a simple ICCBF,* all *Lipschitz continuous controllers $\pi : \mathcal{C}^* \to \mathcal{U}$ render the set $\mathcal{C}^*$ forward invariant.*

Proof. By definition, since b_N is a simple ICCBF, $\mathcal{C}^*$ is a strict subset of $\mathcal{C}_N$. Then $\mathcal{C}^* \cap \partial\mathcal{C}_N = \varnothing$, the null set, i.e., there does not exist a $x \in \mathcal{C}^*$ such that $b_N(x) = 0$. Following Theorem 2.6, we do not need to consider case where $i = N$ in condition (2.33). The remaining cases, with $i \in \{0, ..., N-1\}$ satisfy condition (2.33) for all $\pi(x) \in \mathcal{U}$. Therefore, any Lipschitz continuous $\pi : \mathcal{C}^* \to \mathcal{U}$ admits globally unique solutions and satisfies condition (2.33), completing the proof. □

Intuitively, the existence of a simple ICCBF represents a system where the dynamics at the boundaries of $\mathcal{C}^*$ are such that the unforced dynamics $f(x)$ dominate the forcing term $g(x)\pi(x)$ in driving the system towards safety. If a simple ICCBF is found, no safety critical controller is needed for the system to ensure state trajectories remain within the safe set, provided the system is initialized within $\mathcal{C}^*$.

Remark 2.4. *Higher Order CBFs, as in [176], are a special case of ICCBFs. For instance, in systems of relative degree 2, $L_g h(x) = 0$ for all $x \in \mathcal{S}$. In this case, in the construction of*

ICCBFs we have

$$\begin{aligned} b_1(x) &= \inf_{u\in\mathcal{U}}[L_f h(x) + L_g h(x)u + \alpha_0(h(x))] \\ &= \inf_{u\in\mathcal{U}}[L_f h(x) + \alpha_0(h(x))] \\ &= L_f h(x) + \alpha_0(h(x)) \end{aligned} \tag{2.36}$$

which is exactly the function defined in [176]. This repeats for higher relative degrees. For a system with relative degree ρ, the first ρ expressions of ICCBFs are identical to those of HOCBFs. Moreover, ICCBFs can handle systems with non-uniform relative degree, by choosing N greater or equal to the largest relative degree of the system in $\mathcal{S}$.

Remark 2.5. *The search (over integers N and class $\mathcal{K}$ functions α_i) and validation for ICCBFs (i.e., verifying (2.31)) can be complicated, as is the case with Lyapunov functions in general. For practical implementation, we can solve the following optimization problem:*

$$\gamma = \underset{x\in\mathcal{C}^*}{\textit{minimize}} \quad \sup_{u\in\mathcal{U}}[\dot{b}_N(x,u) + \alpha_N(b_N(x))] \tag{2.37}$$

By the definition, b_N is an ICCBF if and only if the optimization problem is feasible, with solution $\gamma \geq 0$. Since this optimization is nonlinear, unless a guaranteed global optimizer is used, this can only be used to invalidate b_N as a ICCBF. For our experiments, we manually checked a few (approx. 6) N and α_i until $\gamma \geq 0$. Whether a finite N exists for a given dynamical system such that b_N is an ICCBF remains an open question.

2.2.3 Simulations

Adaptive Cruise Control

As a demonstration, we apply ICCBFs to the Adaptive Cruise Control (ACC) problem of [18]. Consider a point-mass model of a vehicle moving in a straight line. The vehicle is following a vehicle d distance in-front, moving at a known constant speed v_0. The objective is to design a controller to accelerate to the speed limit but prevent the vehicles from colliding.

As in [18], the safety constraint is specified as $d \geq 1.8v$. Defining the state $x = [d, v]^T$, the dynamical model is

$$\begin{bmatrix} \dot{d} \\ \dot{v} \end{bmatrix} = \begin{bmatrix} v_0 - v \\ -F(v)/m \end{bmatrix} + \begin{bmatrix} 0 \\ g_0 \end{bmatrix} u,\ \mathcal{U} = \{u : |u| \leq 0.25\}$$

where u is the control input, $F(v) = f_0 + f_1 v + f_2 v^2$ models resistive forces on the vehicle,

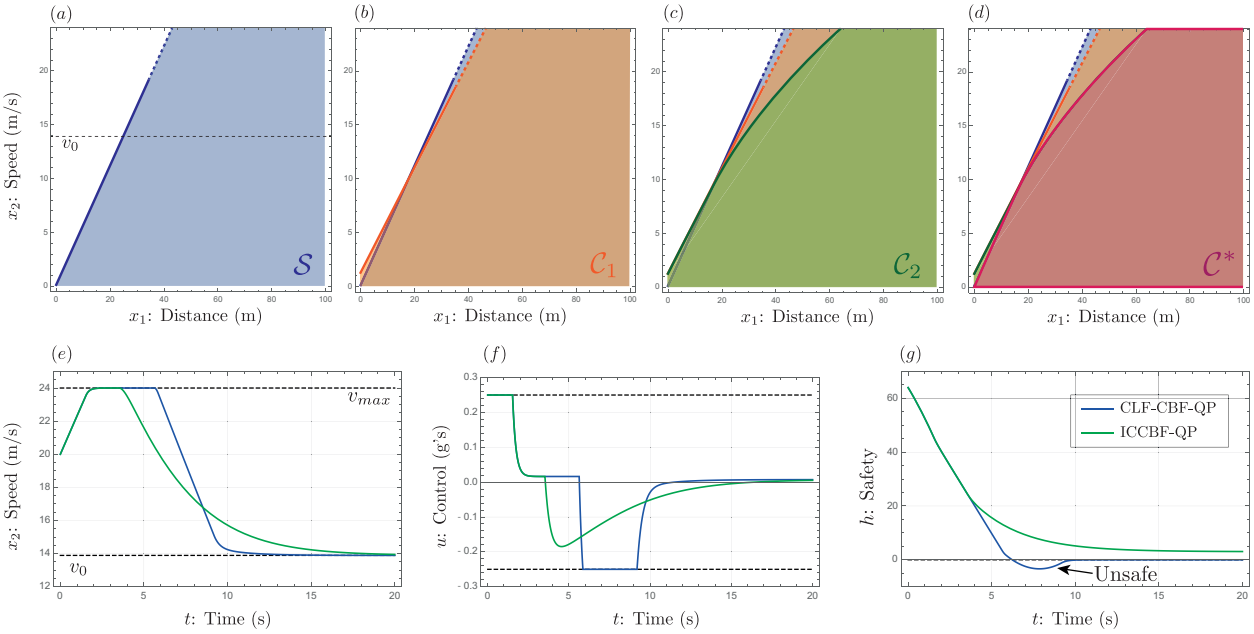

Figure 2.2: Figures (a-d): State-space diagrams indicating the sets (a) $\mathcal{S}$, (b) $\mathcal{C}_1$, (c) $\mathcal{C}_2$ and (d) $\mathcal{C}^*$. The horizontal dashed line in (a) indicates v_0, the speed of the car in-front. Figure (d) represents the inner safe set $\mathcal{C}^*$ that is rendered forward invariant. Figures (e-g): Simulation results for speed, control input and safety under the CLF-CBF-CBF controller [18] and the ICCBF-CBF.

m is the mass of the vehicle, g_0 is acceleration due to the gravity. The safe set $\mathcal{S}$ is

$$\mathcal{S} = \{x \in \mathcal{X} : h(x) = x_1 - 1.8x_2 \geq 0\}$$

and we can verify that $\mathcal{S}$ is not forward invariant under the input constraints. Thus, h is not a CBF, and we will apply ICCBFs to find an inner safe set.

We choose, arbitrarily, $N = 2$ and the class $\mathcal{K}$ functions

$$\alpha_0(h) = 4h, \quad \alpha_1(h) = 7\sqrt{h}, \quad \alpha_2(h) = 2h,$$

to define the functions b_1, b_2 and sets $\mathcal{C}_1, \mathcal{C}_2$. To (approximately) verify that b_2 is an ICCBF, the optimization (2.37) was used, and $\gamma = 2.33$ was found.

The sets are visualized in Figure 2.2. The interior of a set is shaded, and the boundary of the set is indicated with a thick line. Where there exists a feasible control input to keep trajectories within the set, the line is solid, and where no feasible control input will keep trajectories within the set, the line is dashed. $\mathcal{C}^*$, the intersection of $\mathcal{S}, \mathcal{C}_1, \mathcal{C}_2$, is visualized in Figure 2.2(d). The following controller is used:

$$\begin{aligned} \pi(x) = \underset{u \in \mathbb{R}}{\text{argmin}} \quad & \tfrac{1}{2}(u - \pi_d(x))^2 \\ \text{subject to} \quad & L_f b_2(x) + L_g b_2(x) u \geq -2b_2(x) \\ & u \in \mathcal{U} \end{aligned}$$

where $\pi_d(x)$ is the desired acceleration. The desired acceleration is computed using the Control Lyapunov Function $V(x) = (x_2 - v_{max})^2$, where $v_{max} = 24$ is the speed limit. Thus, $\pi_d(x)$:

$$L_f V(x) + L_g V(x) \pi_d(x) = -10V(x)$$

We compare our controller to the CLF-CBF-QP [18]:

$$\begin{aligned} \underset{u \in \mathbb{R}, \delta \in \mathbb{R}_{\geq 0}}{\text{argmin}} \quad & \tfrac{1}{2}u^2 + 0.1\delta^2 \\ \text{subject to} \quad & L_f V(x) + L_g V(x) \leq -10V(x) + \delta \\ & L_f h(x) + L_g h(x) u \geq -2h(x) \end{aligned}$$

and clip of the solutions of the QP such that $u^*(x)$ lies in the range of feasible control inputs.

In Figures 2.2 (e-g), the proposed controller (green) is compared to the CLF-CBF-QP controller (blue). The CLF-CBF-QP reaches the input-constraint at $t = 5.9$ seconds. The

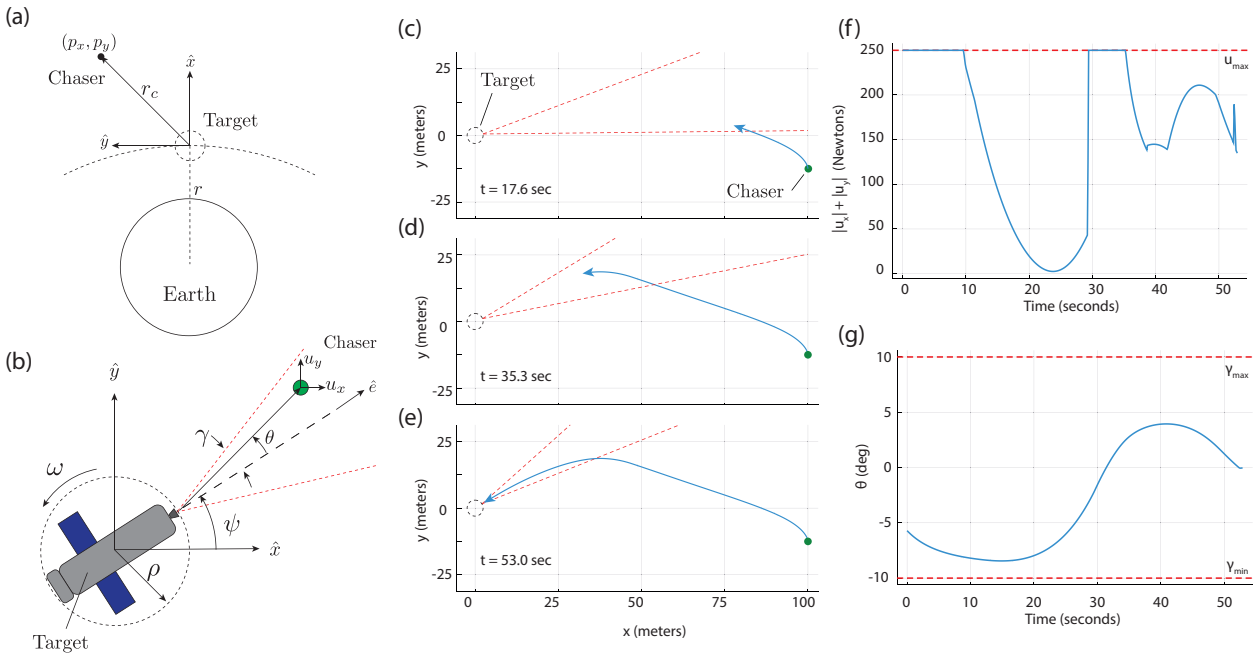

Figure 2.3: (a, b) Schematic of the rendezvous problem. (a) represents the Local-Vertical Local-Horizontal frame. (b) details the target and chaser spacecrafts. The target spacecraft is rotating with constant angular velocity ω. The red dashed lines indicate the Line-of-Sight cone, which the chaser spacecraft must remain within. (c-e) show snapshots of the trajectory at three instances. The green dot represents the initial condition. (f) shows the 1-norm of the propulsive force and (g) indicates the line of sight angle θ.

input limits force the system to leave the safe set. The ICCBF-QP remains feasible and safe for the entire duration, by applying brakes early, at $t = 2.9$ seconds, instead of $t = 5.0$ seconds. Thus, by explicitly accounting for input constraints ICCBF-QP controller keeps the input-constrained system safe, where the CLF-CBF-QP doesn't.

Autonomous Rendezvous

In this section, the ICCBF method is applied to an autonomous rendezvous operation (adapted from [130]) between a chaser spacecraft modeled as a point mass, and a target body, e.g. the International Space Station (ISS) (Figure 2.3). The target is modeled as a point on a disk of radius $\rho = 2.4$ m rotating with a constant angular velocity $\omega = 0.6°/\text{sec}$ relative to the Local-Vertical Local-Horizontal (LVLH) frame. The objective is to determine the appropriate propulsive forces to bring the chaser spacecraft from a range of 100 m to 3 m. The safety constraint is to maintain a line-of-sight (LOS) constraint: the spacecraft's position must remain within a $\gamma = 10°$ cone of the docking axis. The system state $x \in \mathbb{R}^5$ is the relative position (p_x, p_y) and velocity (v_x, v_y) and angle of the docking port ψ. Instead of

using the (linearized) Clohessy-Wiltshire equations (as in [130]), we use the exact equations of relative motion[3]:

$$\frac{d}{dt}\begin{bmatrix} p_x \\ p_y \\ v_x \\ v_y \\ \psi \end{bmatrix} = \begin{bmatrix} v_x \\ v_y \\ n^2 p_x + 2nv_y + \frac{\mu}{r^2} - \frac{\mu(r+p_x)}{r_c^3} \\ n^2 p_y - 2nv_x - \frac{\mu p_y}{r_c^3} \\ \omega \end{bmatrix} + \frac{1}{m_c}\begin{bmatrix} 0 \\ 0 \\ u_x \\ u_y \\ 0 \end{bmatrix} \tag{2.38}$$

where $r_c = \sqrt{x^2 + y^2}$ is the relative distance to the chaser, $r = 6771$ km is the radius of orbit of the ISS, $\mu = 398,600$ km^3/s^2 is the gravitational parameter of Earth, $n = \sqrt{\mu/r^3}$ is the mean motion of the target satellite around the Earth, $\omega = 0.6°$/s is the angular velocity of the target relative to the LVLH frame, and $m_c = 1000$ kg is the mass of the chaser vehicle, assumed constant during the rendezvous. The control inputs (u_x, u_y) are the propulsive forces. Suppose the forces are 1-norm bounded, $|u_x| + |u_y| \leq 0.25$ kN. The LOS constraint is $h(x) \geq 0$, where

$$\begin{aligned} h(x) &= \cos\theta - \cos\gamma \\ &= \frac{\vec{r}_{c-p} \cdot \hat{e}}{||\vec{r}_{c-p}||} - \cos(\gamma), \end{aligned}$$

and $\vec{r}_{c-p} = [(p_x - \rho\cos\psi), (p_y - \rho\sin\psi)]^T$ is the position vector of the chaser relative to the docking port, and $\hat{e} = [\cos\psi, \sin\psi]^T$ is the docking axis vector. We use a CLF to guide the chase to the the docking port:

$$V(x) = \left(v_x + \frac{p_x - \rho\cos\psi}{10}\right)^2 + \left(v_y + \frac{p_y - \rho\sin\psi}{10}\right)^2.$$

To construct the ICCBF, again $N = 2$ was chosen. The following class $\mathcal{K}$ functions were used:

$$\alpha_0(h) = 0.25h, \quad \alpha_1(h) = 0.85h, \quad \alpha_2(h) = (0.05 + k)h$$

where $k > 0$ is a parameter we allow the Quadratic Program to minimize, as in [67], and verified approximately. Thus, the controller u^* is the solution to u in the following quadratic

[3]In this work, only gravitational forces due to the Earth and propulsive forces are modeled, but other nonlinear effects like solar radiation pressure or air resistance can also be included.

optimization problem

$$\begin{aligned} \underset{u\in\mathbb{R}^2;\delta,k\in\mathbb{R}_{\geq 0}}{\text{argmin}} \quad & \tfrac{1}{2}(u_x^2+u_y^2)+10\delta+50k \\ \text{subject to} \quad & L_fV(x)+L_gV(x)u \leq -0.1V(x)+\delta \\ & L_fb_2(x)+L_gb_2(x)u \geq -(0.05+k)b_2(x) \\ & |u_x|+|u_y| \leq 0.25 \end{aligned}$$

Figure 2.3(c-g) show simulation results of the rendezvous operation. The chaser is initialized at (100, -10) meters from the target spacecraft, and follows the trajectories drawn in (c-e), demonstrating a successful transfer. The 1-norm of the computed thrust force is indicated in (f), and (g) shows that the LOS constraint is satisfied at all times during the transfer. 3D animations, videos and source code for both case studies are available at https://github.com/dev10110/Input-Constrained-Control-Barrier-Functions.

2.2.4 Conclusion

In this section we have presented a framework that allows input constraints to be explicitly included in the construction of control barrier functions and to guarantee that safety is maintained with an input-constrained controller. The construction identifies an inner safe set and a feedback controller to render the subset safe. We demonstrated the method on an adaptive cruise control problem and a spacecraft rendezvous problem. An optimization based method was used to verify the conditions of the ICCBF. Directions for future work include investigating numerically efficient methods to automate the search of ICCBFs, and to compare the complexity with other reachability methods, in particular for systems with high-dimensional states. Finally, the robustness of this controller to noise and model mismatch could also be investigated.

2.3 Observer-Controller Interconnections

In recent years, Control Barrier Functions (CBFs) [21] have become a popular method to design safety-critical controllers, since a certifiably safe control input can be computed efficiently for nonlinear systems. Many extensions have been proposed to address specific challenges in using CBFs, including robustness [13, 84], sampled-data considerations [36] and integration with high-level planners [8]. However, these works assume the controller has access to perfect state information. In most practical systems, the true state of the system is unknown and must be reconstructed using only (often noisy) measurements obtained from sensors. In such systems, it is common to design a full-state feedback controller, and then replace the state by an estimate provided by an observer [30, Sec. 8.7]. It is well established that a controller capable of stabilizing a system with perfect state information may fail to do so when using the state estimate [91, Ch. 12]. Similarly, the use of imperfect information for feedback control may cause safety violations.

Here, we study the implications on safety that arise due to imperfect and partially available information, and propose a method to design safe observer-controllers. This important challenge has only recently received some attention. Measurement-Robust CBFs [57] have been proposed to address control synthesis in output-feedback, in the context of vision-based control. The authors assume sensors are noiseless and an imperfect inverse of the measurement map is known, i.e. from a single measurement, a ball containing the true state is known. Using this bound, a second-order cone program-based controller was proposed, although the Lipschitz continuity of this controller is yet to be established [57]. For many safety-critical systems, the measurement maps are non-invertible, limiting the scope for this method.

In [51], a safety critical controller is proposed for stochastic systems, and a probabilistic safety guarantee is proved. The authors consider linear (non-invertible) measurement maps, additive gaussian disturbances, and specifically use the Extended Kalman Filter (EKF) as the observer. In [83] this work is extended to consider a broader class of control-affine systems, and probabilistic guarantees of safety over a finite forward interval are obtained. Establishing safety in a deterministic (non-probabilistic) sense or using alternative observers remains challenging. It has also been demonstrated that in some cases, safety guarantees can be obtained by modeling the system as a Partially Observable Markov Decision Process, e.g. [11], although such methods are computationally expensive for high-dimensional systems and are more suitable for systems with discrete action/state spaces.

The primary contribution of this section is in synthesizing safe and robust interconnected observer-controllers in such a manner as to establish rigorous guarantees of safety, despite bounded disturbances on the system dynamics and sensor measurements. We propose two

approaches to solve this problem, owing to the wide range of nonlinear observers [30]. The first approach utilizes the class of Input-to-State Stable observers [150]. The second approach employs the more general class of 'Bounded Error' observers, in which a set containing the state estimation error is known at all times. This class of observers includes the Deterministic Extended Kalman Filter (DEKF) [91, Ch. 11.2], Lyapunov-based sum-of-squares polynomial observers [133], and others discussed later. We show that our safe estimate-feedback controller can be obtained by solving quadratic programs (QP), and prove Lipschitz continuity of these controllers, allowing for low-computational complexity real-time implementation. The efficacy of the methods is demonstrated both in simulations and in experiments on a quadrotor.

2.3.1 Preliminaries and Background

In this section, let γ_f denote the Lipschitz constant of a Lipschitz-continuous function $f : \mathbb{R}^n \to \mathbb{R}^m$.

System

Consider a nonlinear control-affine system:

$$\dot{x} = f(x) + g(x)u + g_d(x)d(t), \tag{2.39a}$$

$$y = c(x) + c_d(x)v(t), \tag{2.39b}$$

where $x \in \mathcal{X} \subset \mathbb{R}^n$ is the system state, $u \in \mathcal{U} \subset \mathbb{R}^m$ is the control input, $y \in \mathbb{R}^{n_y}$ is the measured output, $d : \mathbb{R}_{\geq 0} \to \mathbb{R}^{n_d}$ is a disturbance on the system dynamics, and $v : \mathbb{R}_{\geq 0} \to \mathbb{R}^{n_v}$ is the measurement disturbance. We assume d and v are piecewise continuous, bounded disturbances, $\sup_t \|d(t)\|_\infty = \bar{d}$ and $\|v(t)\|_\infty \leq \bar{v}$ for some known $\bar{d}, \bar{v} < \infty$. The functions $f : \mathcal{X} \to \mathbb{R}^n$, $g : \mathcal{X} \to \mathbb{R}^{n \times m}$, $c : \mathcal{X} \to \mathbb{R}^{n_y}$, $g_d : \mathcal{X} \to \mathbb{R}^{n \times n_d}$, and $c_d : \mathcal{X} \to \mathbb{R}^{n_y \times n_v}$ are all assumed to be locally Lipschitz continuous. Notice that $g_d(x)d(t)$ accounts for either matched or unmatched disturbances.

In observer-controller interconnections, the observer maintains a state estimate $\hat{x} \in \mathcal{X}$, from which the controller determines the control input. The observer-controller interconnection is defined to be of the form:

$$\dot{\hat{x}} = p(\hat{x}, y) + q(\hat{x}, y)u, \tag{2.40a}$$

$$u = \pi(t, \hat{x}, y), \tag{2.40b}$$

where $p : \mathcal{X} \times \mathbb{R}^{n_y} \to \mathbb{R}^n$, $q : \mathcal{X} \times \mathbb{R}^{n_y} \to \mathbb{R}^{n \times m}$ are locally Lipschitz in both arguments. The feedback controller $\pi : \mathbb{R}_{\geq 0} \times \mathcal{X} \times \mathbb{R}^p \to \mathcal{U}$ is assumed piecewise-continuous in t and Lipschitz continuous in the other two arguments. Then, the closed-loop system formed by (2.39, 2.40) is

$$\dot{x} = f(x) + g(x)u + g_d(x)d(t), \tag{2.41a}$$

$$\dot{\hat{x}} = p(\hat{x}, y) + q(\hat{x}, y)u, \tag{2.41b}$$

$$x(0) = x_0, \ \hat{x}(0) = \hat{x}_0, \tag{2.41c}$$

where y and u are defined in (2.39b) and (2.40b) respectively. Under the stated assumptions, there exists an interval $\mathcal{I} = \mathcal{I}(x_0, \hat{x}_0) = [0, t_{max}(x_0, \hat{x}_0))$ over which solutions to the closed-loop system exist and are unique [92, Thm 3.1].

Safety

Safety is defined as the true state of the system remaining within a *safe set*, $\mathcal{S} \subset \mathcal{X}$, for all times $t \in \mathcal{I}$. The safe set $\mathcal{S}$ is defined as the super-level set of a continuously-differentiable function $h : \mathcal{X} \to \mathbb{R}$, as in (2.10):

$$\mathcal{S} = \{x \in \mathcal{X} : h(x) \geq 0\}. \tag{2.42}$$

A state-feedback controller[4] $\pi : \mathbb{R}_{\geq 0} \times \mathcal{X} \to \mathcal{U}$ *renders system* (2.39) *safe* with respect to the set $\mathcal{S}$, if for the closed-loop dynamics $\dot{x} = f(x) + g(x)\pi(t, x) + g_d(x)d(t)$, the set $\mathcal{S}$ is *forward invariant*, i.e., $x(0) \in \mathcal{S} \implies x(t) \in \mathcal{S} \ \forall t \in \mathcal{I}$. In output-feedback we define safety as follows:

Definition 2.14. *An observer-controller pair* (2.40) ***renders system*** (2.39) ***safe*** *with respect to a set $\mathcal{S} \subset \mathcal{X}$ from the initial-condition sets $\mathcal{X}_0, \hat{\mathcal{X}}_0 \subset \mathcal{S}$ if for the closed-loop system* (2.41)*,*

$$x(0) \in \mathcal{X}_0 \text{ and } \hat{x}(0) \in \hat{\mathcal{X}}_0 \implies x(t) \in \mathcal{S} \quad \forall t \in \mathcal{I}. \tag{2.43}$$

Note the importance of the observer-controller connection, i.e., using only $\hat{x}(t)$, we must obtain guarantees on $x(t)$.

[4]In *state-feedback* the control input is determined from the true state, $u = \pi(t, x)$. In *estimate-feedback* the input is determined from the state estimate and measurements, $u = \pi(t, \hat{x}, y)$.

Control Barrier Functions

Control Barrier Functions (CBFs) have emerged as a tool to characterize and find controllers that can render a system safe [21]. Robust-CBFs [84] also account for the disturbances $d(t)$ in (2.39a). We introduce a modification to reduce conservatism, inspired by [13].

Definition 2.15. *A continuously differentiable function $h : \mathcal{X} \to \mathbb{R}$ is a **Tunable Robust CBF** (TRCBF) for system (2.39) if there exists a class $\mathcal{K}$ function α, and a continuous, non-increasing function $\kappa : \mathbb{R}_{\geq 0} \to \mathbb{R}$ with $\kappa(0) = 1$, s.t.*

$$\sup_{u \in \mathcal{U}} L_f h(x) + L_g h(x) u + \alpha(h(x)) \geq \kappa(h(x)) \left\| L_{g_d} h(x) \right\| \bar{d}, \ \forall x \in \mathcal{S}. \tag{2.44}$$

Examples include $\kappa(r) = 1$ and $\kappa(r) = 2/(1 + \exp(r))$. Given a TRCBF h for (2.39), the set of safe control inputs is

$$K_{trcbf}(x) = \{u \in \mathcal{U} : \ L_f h(x) + L_g h(x) u - \kappa(h(x)) \left\| L_{g_d} h(x) \right\| \bar{d} \geq -\alpha(h(x))\}, \tag{2.45}$$

and a safe state-feedback controller is obtained by solving a QP, as in [84, Eq. 30]. The main question is:

Problem 2.2. *Given a system (2.39) with disturbances of known bounds $\|d(t)\|_\infty \leq \bar{d}$, $\|v(t)\|_\infty \leq \bar{v}$, and a safe set $\mathcal{S}$ defined by (2.42), synthesize an interconnected observer-controller (2.40) and the initial condition sets $\mathcal{X}_0, \hat{\mathcal{X}}_0$ to render the system safe.*

We study systems subject to disturbances with a known bound. We will use this bound to derive sufficient conditions on the control policy to guarantee safety satisfaction. In practice, a conservative upper bounds can be used, although future work will address the probabilistic safety guarantees that are possible under probabilistic disturbances.

2.3.2 Main Results

2.3.2.1 Approach 1

Approach 1 relies on defining a set of state estimates, $\hat{\mathcal{S}} \subset \mathcal{X}$, such that if the estimate $\hat{x}$ lies in $\hat{\mathcal{S}}$, the true state x lies in the safe set $\mathcal{S}$. The controller is designed to ensure $\hat{x} \in \hat{\mathcal{S}}$ at all times. We consider Input-to-State Stable observers:

Definition 2.16 (Adapted from [150])**.** *An observer (2.40) is an **Input-to-State (ISS) Observer** for system (2.39), if there exists a class $\mathcal{KL}$ function β continuously differentiable*

wrt to the second argument, and a class $\mathcal{K}$ function η such that

$$\|x(t) - \hat{x}(t)\| \leq \beta(\|x(0) - \hat{x}(0)\|, t) + \eta(\bar{w}), \forall t \in \mathcal{I}, \tag{2.46}$$

where $\bar{w} = \max(\bar{d}, \bar{v})$.

Various methods to design ISS observers for nonlinear systems have been developed, and reader is referred to [14, 23, 30, 81, 150] and references within for specific techniques.

The key property of an ISS observer is that the estimation error is bounded with a known bound: for any $\delta > 0$, there exists a continuously differentiable, non-increasing function $M_\delta : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$, such that

$$\|x(0) - \hat{x}(0)\| \leq \delta \Rightarrow \|x(t) - \hat{x}(t)\| \leq M_\delta(t) \ \forall t \in \mathcal{I}. \tag{2.47}$$

Comparing (2.46) and (2.47), $M_\delta(t) = \beta(\delta, t) + \eta(\bar{w})$. Define

$$\hat{\mathcal{S}} = \{\hat{x} \in \mathcal{X} : h(\hat{x}) - \gamma_h M_\delta(t) \geq 0\}, \tag{2.48}$$

the set of safe state-estimates, and we obtain the property $\hat{x}(t) \in \hat{\mathcal{S}} \implies x(t) \in \mathcal{S}$ by the Lipschitz continuity of h.[5] Then the conditions to guarantee safety are as follows:

Definition 2.17. *A continuously differentiable function $h : \mathcal{X} \to \mathbb{R}$ is an **Observer-Robust CBF** for system (2.39) with an ISS observer (2.40a) of known estimation error bound (2.47), if there exists an extended class $\mathcal{K}$ function α s.t.*[6]

$$\sup_{u \in \mathcal{U}} L_p h(\hat{x}, y) + L_q h(\hat{x}, y) u \geq -\alpha(h(\hat{x}) - \gamma_h M_\delta(0)) \tag{2.49}$$

for all $\hat{x} \in \mathcal{S}$, and all $y \in \mathcal{Y}(\hat{x}) = \{y : y = c(x) + c_d(x)v(t) \mid \|x - \hat{x}\| \leq M_\delta(0), \|v\| \leq \bar{v}\}$, an overapproximation of the set of possible outputs.[7]

Theorem 2.8. *For system (2.39), suppose the observer (2.40a) is ISS with estimation error bound (2.47). Suppose $\mathcal{S}$ is defined by an Observer-Robust CBF $h : \mathcal{X} \to \mathbb{R}$ associated with*

[5] By Lipschitz continuity, $|h(x) - h(\hat{x})| \leq \gamma_h \|x - \hat{x}\| \implies h(\hat{x}) - \gamma_h \|x - \hat{x}\| \leq h(x)$. Therefore, if $\hat{x} \in \hat{\mathcal{S}}$, then $0 \leq h(\hat{x}) - \gamma_h M_\delta(t) \leq h(\hat{x}) - \gamma_h \|x - \hat{x}\| \leq h(x)$, i.e., $x \in \mathcal{S}$. Thus, $\hat{x} \in \hat{\mathcal{S}} \implies x \in \mathcal{S}$.

[6] Recall the notation $L_p h(\hat{x}, y) = \frac{\partial h}{\partial x}(\hat{x}) p(\hat{x}, y)$.

[7] $\mathcal{Y}$ is defined using $M_\delta(0)$ instead of δ since $\mathcal{Y}(\hat{x}(t))$ must contain the set of possible outputs at time t for all $t \in \mathcal{I}$.

extended class $\mathcal{K}$ function α. If the initial conditions satisfy

$$\hat{x}(0) \in \hat{\mathcal{X}}_0 = \{\hat{x} \in \mathcal{S} : h(\hat{x}) \geq \gamma_h M_\delta(0)\}, \tag{2.50}$$

$$x(0) \in \mathcal{X}_0 = \{x \in \mathcal{S} : \|x(0) - \hat{x}(0)\| \leq \delta\}, \tag{2.51}$$

then any Lipschitz continuous estimate-feedback controller $u = \pi(t, \hat{x}, y) \in K_{orcbf}(t, \hat{x}, y)$ where

$$K_{orcbf}(t, \hat{x}, y) = \{u \in \mathcal{U} : L_p h(\hat{x}, y) + L_q h(\hat{x}, y)u \geq -\alpha\left(h(\hat{x}) - \gamma_h M_\delta(t)\right) + \gamma_h \dot{M}_\delta(t)\} \tag{2.52}$$

renders the system safe from the initial-condition sets $\mathcal{X}_0, \hat{\mathcal{X}}_0$.

Proof. Consider the function $H(t, \hat{x}) = h(\hat{x}) - \gamma_h M_\delta(t)$. By the Lipschitz continuity of h, and (2.47), $H(t, \hat{x}) \geq 0 \implies h(x) \geq 0$. The total derivative of H is

$$\dot{H} = \frac{\partial H}{\partial t} + \frac{\partial H}{\partial \hat{x}}\dot{\hat{x}} = -\gamma_h \dot{M}_\delta + L_p h(\hat{x}, y) + L_q h(\hat{x}, y)u$$

therefore, for any $\pi(t, \hat{x}, y) \in K_{orcbf}(t, \hat{x}, y)$ we have $\dot{H} \geq -\alpha(H)$. Since $H(0, \hat{x}_0) \geq 0$ (from the initial condition (2.50)), $H(t, \hat{x}) \geq 0, \forall t \in \mathcal{I}$, completing the proof. □

Remark 2.6. *Under the same assumptions as Theorem 2.8, if $\mathcal{U} = \mathbb{R}^m$ and a desired control input $\pi_{des} : \mathbb{R}_{\geq 0} \times \mathcal{X} \to \mathbb{R}^m$ is provided, a QP-based safe estimate-feedback controller is*

$$\pi(t, \hat{x}, y) = \underset{u \in \mathbb{R}^m}{\operatorname{argmin}} \ \|u - \pi_{des}(t, \hat{x})\|^2 \tag{2.53a}$$

$$\text{subject to } L_p h(\hat{x}, y) + L_q h(\hat{x}, y)u \geq -\alpha(h(\hat{x}) - \gamma_h M_\delta(t)) + \gamma_h \dot{M}_\delta(t) \tag{2.53b}$$

Remark 2.7. *The constraint in (2.53) does not explicitly depend on the disturbances $d(t)$ and $v(t)$, since the effect of these disturbances is captured by the estimation error bound $M_\delta(t)$. Furthermore, since $\gamma_h \dot{M}_\delta(t) \leq 0$,[8] the constraint (2.53) is easier to satisfy for higher convergence rates of the observer.*

Remark 2.8. *For a linear class $\mathcal{K}$ function, $\alpha(r) = \gamma_\alpha r$, if $\dot{M}_\delta \leq -\gamma_\alpha M_\delta(t)$, a sufficient condition for (2.53) is*

$$L_p h(\hat{x}, y) + L_q h(\hat{x}, y)u \geq -\gamma_\alpha h(\hat{x}).$$

[8] Since $M_\delta(t) = \beta(\delta, t) + \eta(\bar{w})$, and β is a class $\mathcal{KL}$ function, $\dot{M}_\delta(t) = \partial\beta/\partial t < 0$. Finally since $\gamma_h \in \mathbb{R}_{\geq 0}$ is a Lipschitz constant, $\gamma_h \dot{M}_\delta(t) \leq 0$.

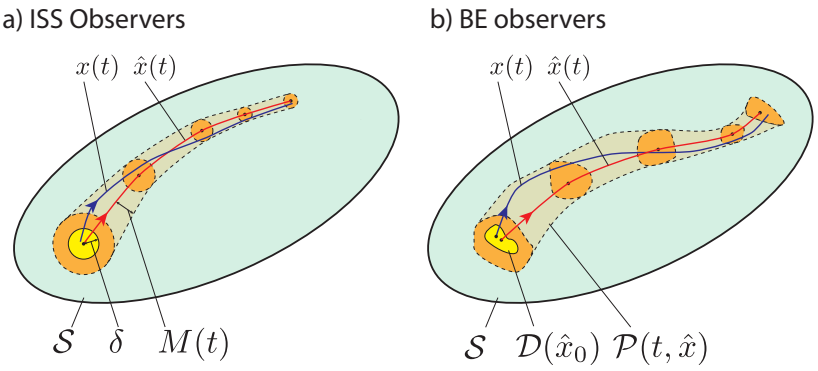

Figure 2.4: Depiction of Input-to-State Stable observers and Bounded-Error observers. (a) In ISS observers, the estimation error is bounded by a norm-ball, and must be non-increasing in time. (b) In BE observers, the state estimate must be contained in a bounded set $\mathcal{P}(t,\hat{x})$.

which does not depend on the bound $M_\delta(t)$ or Lipschitz constant γ_h. In other words, if the observer converges faster than the rate at which the boundary of the safe set is approached, i.e., if $\dot{M}_\delta \leq -\gamma_\alpha M_\delta$, then a safe control input can be obtained without explicit knowledge of M_δ or γ_h. This matches the general principle that for good performance observers should be converge faster than controllers.

2.3.2.2 Approach 2

While in Approach 1 we used the stability guarantees of ISS observers to obtain safe controllers, in Approach 2 we consider observers that only guarantee boundedness of the estimation error. First, we define Bounded-Error Observers:

Definition 2.18. *An observer* (2.40a) *is a* ***Bounded-Error (BE) Observer****, if there exists a bounded set $\mathcal{D}(\hat{x}_0) \subset \mathcal{X}$ and a (potentially) time-varying bounded set $\mathcal{P}(t,\hat{x}) \subset \mathcal{X}$ s.t.*

$$x_0 \in \mathcal{D}(\hat{x}_0) \implies x(t) \in \mathcal{P}(t,\hat{x}) \ \forall t \in \mathcal{I}. \tag{2.54}$$

Figure 2.4 depicts the sets $\mathcal{D}$ and $\mathcal{P}$. Note, ISS observers are a subset of BE observers, using the definitions $\mathcal{D}(\hat{x}_0) = \{x : \|x - \hat{x}_0\| \leq \delta\}$ and $\mathcal{P}(t,\hat{x}) = \{x : \|x - \hat{x}(t)\| \leq M_\delta(t)\}$. BE observers are more general than ISS observers in the following ways: (A) The sets $\mathcal{D}$ and $\mathcal{P}$ do not have to be norm-balls. For example, they could be zonotopes [12], intervals [85], or sublevel sets of sum-of-squares polynomials [15]. (B) The shape and size of $\mathcal{P}$ is allowed to change over time.

The idea is to find a common, safe input for all $x \in \mathcal{P}(t,\hat{x})$:

Theorem 2.9. *For system* (2.39)*, suppose the observer* (2.40a) *is a Bounded-Error observer. Suppose the safe set $\mathcal{S}$ is defined by a continuously differentiable function $h : \mathcal{X} \to \mathbb{R}$, where*

h is a Tunable Robust-CBF for the system. Suppose $\pi : \mathbb{R}_{\geq 0} \times \mathcal{X} \to \mathcal{U}$ is an estimate-feedback controller, piecewise-continuous in the first argument and Lipschitz continuous in the second, s.t.

$$\pi(t, \hat{x}) \in \bigcap_{x \in \mathcal{P}(t,\hat{x})} K_{trcbf}(x), \tag{2.55}$$

where K_{trcbf} is defined in (2.45). Then the observer-controller renders the system safe from the initial-condition sets $x(0) \in \mathcal{X}_0 = \mathcal{D}(\hat{x}_0)$ and $\hat{x}_0 \in \hat{\mathcal{X}}_0 = \{\hat{x} : \mathcal{P}(0, \hat{x}_0) \subset \mathcal{S}\}$.

Proof. The total derivative of h for any $x \in \partial\mathcal{S}$ and $\pi(t, \hat{x}) \in K_{trcbf}(x)$ satisfies

$$\begin{aligned}
\dot{h} &= L_f h(x) + L_g h(x)\pi(t, \hat{x}) + L_{g_d} h(x) w(t) \\
&\geq L_f h(x) + L_g h(x)\pi(t, \hat{x}) - \kappa(0) \left\| L_{g_d} h(x) \right\| \bar{w} \\
&\geq -\alpha(0) = 0
\end{aligned}$$

since $h(x) = 0$, $\kappa(0) = 1$, and $x(t) \in \mathcal{P}(t, \hat{x})$. Therefore, at any $x \in \partial\mathcal{S}$, $\dot{h} \geq 0$, i.e., the system remains safe [31]. □

In general, designing a controller satisfying (2.55) can be difficult. We propose a method under the following assumptions:

Assumption 2.1. *There exists a known function $a : \mathbb{R}_{\geq 0} \times \mathcal{X} \to \mathbb{R}$, piecewise continuous in the first argument and Lipschitz continuous in the second, such that for all $\hat{x} \in \mathcal{S}$,*

$$a(t, \hat{x}) \leq \inf_{x \in \mathcal{P}(t,\hat{x})} L_f h(x) - \kappa(h(x)) \left\| L_{g_d} h(x) \right\| \bar{w} + \alpha(h(x)).$$

By Assumption 1, $a(t, \hat{x})$ lower-bounds the terms in $\dot{h}$ independent of u. These bounds can be obtained using Lipschitz constants. Similarly, we bound each term of $L_g h$:

Assumption 2.2. *There exist known functions $b_i^-, b_i^+ : \mathbb{R}_{\geq 0} \times \mathcal{X} \to \mathbb{R}$ for $i = \{1, ..., m\}$, piecewise continuous in the first argument and Lipschitz continuous in the second, such that*[9]

$$b_i^-(t, \hat{x}) \leq [L_g h(x)]_i \leq b_i^+(t, \hat{x})$$

for all $t \geq 0$, all $x \in \mathcal{S}$ and all $\hat{x} \in \{\hat{x} : x \in \mathcal{P}(t, \hat{x})\}$. Furthermore, suppose $\text{sign}(b_i^-(t, \hat{x})) = \text{sign}(b_i^+(t, \hat{x}))$ at every $t, \hat{x} \in \mathcal{S}$, and that h is of relative-degree 1, i.e., $L_g h(x) \neq 0$.

[9]Recall, $[L_g h(x)]_i$ refers to the i-th element of $L_g h(x)$.

Intuitively, by assuming $\text{sign}(b_i^-(t,\hat{x})) = \text{sign}(b_i^+(t,\hat{x}))$ it is clear whether a positive or negative u_i increases $\dot{h}(x,u)$.[10]

Theorem 2.10. *Consider a system* (2.39) *with* $\mathcal{U} = \mathbb{R}^m$ *and suppose the observer* (2.40a) *is a Bounded-Error observer. Suppose* $\mathcal{S}$ *is the safe set defined by an TRCBF* h *and Assumptions 2.1, 2.2 are satisfied. Suppose* $\pi_{des} : \mathbb{R}_{\geq 0} \times \mathcal{X} \to \mathcal{U}$ *is a desired controller, piecewise continuous wrt* t *and Lipschitz continuous wrt* $\hat{x}$*. Then the estimate-feedback controller* $\pi : \mathbb{R}_{\geq 0} \times \mathcal{X} \to \mathbb{R}^m$

$$\pi(t,\hat{x}) = \underset{u\in\mathbb{R}^m}{\text{argmin}} \ \|u - \pi_{des}(t,\hat{x})\|^2 \tag{2.56a}$$

$$\text{subject to } a(t,\hat{x}) + \sum_{i=1}^{m} \min\{b_i^-(t,\hat{x})u_i, b_i^+(t,\hat{x})u_i\} \geq 0 \tag{2.56b}$$

is piecewise continuous wrt t*, Lipschitz continuous wrt* x*, and renders the system safe from the initial-condition sets* $x_0 \in \mathcal{X}_0 = \mathcal{D}(\hat{x}_0)$ *and* $\hat{x}_0 \in \hat{\mathcal{X}}_0 = \{\hat{x} : \mathcal{P}(0,\hat{x}_0) \subset \mathcal{S}\}$*.*

Proof. First, we prove existence and uniqueness of solutions to the QP. In standard form, the QP (2.56) is equivalent to

$$\min_{u\in\mathbb{R}^m, k\in\mathbb{R}^m} \ \frac{1}{2}u^T u - \pi_{des}^T u \tag{2.57a}$$

$$\text{s.t.} \left[\begin{array}{ccc|ccc} b_1^- & \dots & 0 & -1 & \dots & 0 \\ b_1^+ & \dots & 0 & -1 & \dots & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & \dots & b_m^- & 0 & \dots & -1 \\ 0 & \dots & b_m^+ & 0 & \dots & -1 \\ \hline 0 & \dots & 0 & 1 & \dots & 1 \end{array}\right] \left[\begin{array}{c} u_1 \\ \vdots \\ u_m \\ \hline k_1 \\ \vdots \\ k_m \end{array}\right] \geq \left[\begin{array}{c} 0 \\ 0 \\ \vdots \\ 0 \\ 0 \\ \hline -a \end{array}\right] \tag{2.57b}$$

where the dependencies on $(t,\hat{x})$ were omitted for brevity. Here $k \in \mathbb{R}^m$ is an auxiliary variable encoding the constraint $k_i \leq \min\{b_i^- u_i, b_i^+ u_i\}$ for all $i = \{1,...,m\}$. This constraint matrix has size $(2m+1, 2m)$. However, since $\text{sign}(b_i^-) = \text{sign}(b_i^+)$ by Assumption 2.2, only one of either the $(2i-1)$-th or $(2i)$-th constraints can be active.[11] Considering the sparsity pattern of active constraint matrix, these constraints must be linearly independent. Therefore, the proposed QP has $2m$ decision variables with at most $m+1$ linearly independent constraints,

[10] Future work will attempt to relax this assumption. In our limited experience, the estimation error can be sufficiently small that the assumption holds.

[11] Note, if $b_i^- = b_i^+ \neq 0$, then both constraints are equivalent, and thus still means a single constraints is active. Since $L_g h(x) \neq 0$ (Assumption 2.2), $b_i^- = b_i^+ \neq 0$ for at least one of $i = 1,...,m$.

and thus a non-empty set of feasible solutions. Since the cost function is quadratic, there exists a unique minimizer.

Second, we prove Lipschitz continuity. Since the active constraints matrix has linearly independent rows, the regularity conditions in [75] are met. Thus the solution $\pi(t, \hat{x})$ is Lipschitz continuous wrt $\pi_{des}(t, \hat{x}), a(t, \hat{x}), b_i^-(t, \hat{x})$ and $b_i^+(t, \hat{x})$. Since these quantities are piecewise continuous wrt t and Lipschitz continuous wrt $\hat{x}$, the same is true for $\pi(t, \hat{x})$.

Finally, we prove safety. Since (omitting $t, x, \hat{x}$),

$$L_g h u = \sum_{i=1}^{m} [L_g h]_i u_i \geq \sum_{i=1}^{m} \min\{b_i^- u_i, b_i^+ u_i\},$$

satisfaction of the constraint in (2.56) implies satisfaction of (2.55). Therefore, by Theorem 2.9, the system is rendered safe. □

2.3.3 Simulations and Experiments

Code and videos are available here: https://github.com/dev10110/robust-safe-observer-controllers

Simulation: Double Integrator

We simulate a double integrator system without disturbances, to demonstrate the importance of the observer-controller interconnection. The system is (with $\mathcal{U} = \mathbb{R}$)

$$\dot{x}_1 = x_2, \ \dot{x}_2 = u, \ y = x_1, \tag{2.58}$$

and the safe set is defined as $\mathcal{S} = \{x : x_1 \leq x_{max}\}$. We use the CBF $h(x) = -x_2 + \alpha_0(x_{max} - x_1)$. A Luenberger-observer, $\dot{\hat{x}} = A\hat{x} + Bu + L(y - C\hat{x})$, is used, where $L = 1/2 P^{-1} C^T$ and $P \in \mathbb{S}^2_{++}$ is the solution the Lyapunov equation $PA + A^T P - C^T C = -2\theta P$ for design parameter $\theta > 0$. This observer is ISS, since for any $\delta > 0$, (2.47) is satisfied with $M_\delta(t) = \sqrt{\lambda_{max}(P)/\lambda_{min}(P)}\delta e^{-\theta t}$. This observer is also a Bounded Error observer since for any $\delta > 0$, (2.54) is satisfied with $\mathcal{D}(\hat{x}_0) = \{x : \|x_0 - \hat{x}_0\| \leq \delta\}$ and $\mathcal{P}(t, \hat{x}) = \{x : (x - \hat{x})^T P (x - \hat{x}) \leq \lambda_{max}(P)\delta^2 e^{-2\theta t}\}$.

We compare the methods proposed here to the CBF-QP of [21] (referred to as the Baseline-QP), using $\hat{x}$ in lieu of x. Plots of the resulting trajectory are depicted in Figure 2.7, demonstrating safety violation. The trajectory plots under the controllers based on Approaches 1 and 2 are shown in Figure 2.5, demonstrating that safety is maintained in both cases. In Approach 2, the function $L_f h(x)$ is affine in x and $L_g h(x) = -1$ is independent of x,

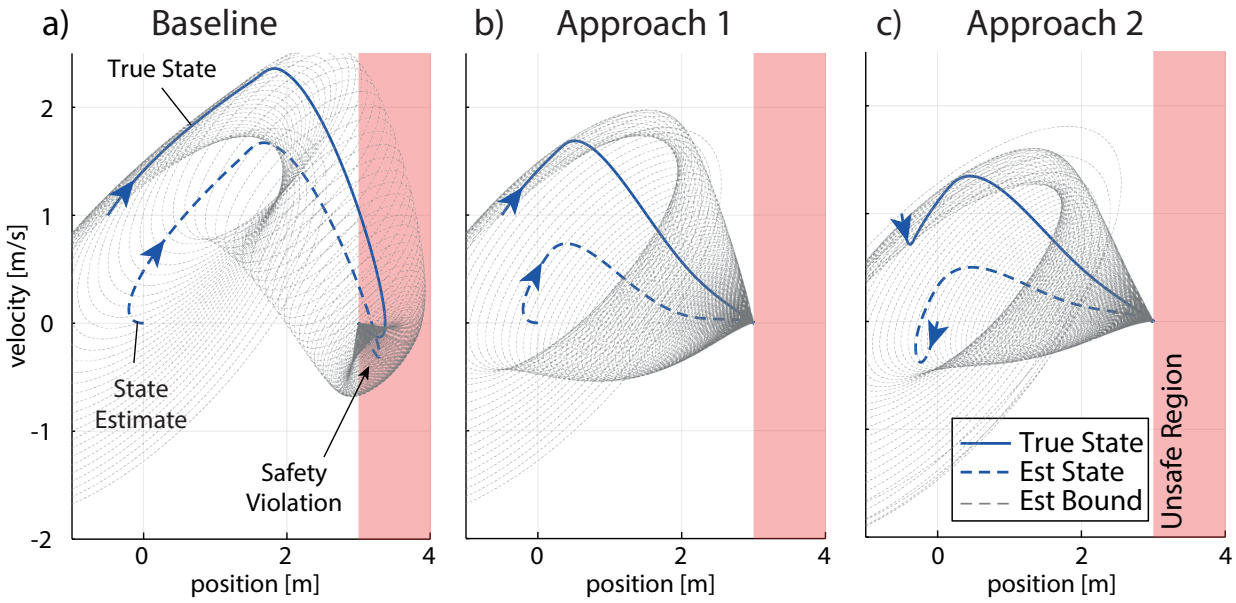

Figure 2.5: Simulation results for the Double Integrator (2.58), using (a) the baseline CBF controller, (b) Approach 1 and (c) Approach 2. The same initial conditions and observer is used for each simulation.

and therefore the function $a(t, \hat{x})$ was determined using a box bound around $\mathcal{P}(t, \hat{x})$ and $b_i^-(t, \hat{x}) = b_i^+(t, \hat{x}) = -1$. Numerically, we have noticed that for some initial conditions and convergence rates, the controller of Approach 1 is less conservative than the controller of Approach 2, and in other cases the converse is true. Identifying conditions that determine whether Approach 1 or 2 is less conservative remains an open question.

Simulation: Planar Quadrotor

Consider

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \\ \dot{x}_5 \\ \dot{x}_6 \end{bmatrix} = \begin{bmatrix} x_4 \\ x_5 \\ x_6 \\ 0 \\ -g \\ 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ \sin x_3/m & 0 \\ \cos x_3/m & 0 \\ 0 & J^{-1} \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ d_1(t) \\ d_2(t) \\ 0 \end{bmatrix}$$
$$y = \begin{bmatrix} x_1, x_2, x_3 \end{bmatrix}^T + \begin{bmatrix} v_1(t), v_2(t), v_3(t) \end{bmatrix}^T$$

where $[x_1, x_2]^T$ are the position coordinates of the quadrotor with respect to an inertial coordinate frame, x_3 is the pitch angle, $[x_4, x_5]^T$ are the linear velocities in the inertial

frame, and x_6 is the rate of change of pitch. The quadrotor has mass $m = 1.0$ kg and moment of inertia $J = 0.25$ kg/m^2, and the acceleration due to the gravity is $g = 9.81$ m/s^2. The control inputs are thrust u_1 and torque u_2. The disturbances $d : \mathbb{R}_{\geq 0} \to \mathbb{R}^2$ captures the effect of unmodeled aerodynamic forces on the system, bounded by $\|d\| \leq 2$ m/s^2. The measurement disturbance is $v : \mathbb{R}_{\geq 0} \to \mathbb{R}^3$, bounded by 5 cm for position measurements, and 5° for pitch measurements.

The safety condition is to avoid collision with a circular obstacle at $[x_1^*, x_2^*]^T$ of radius r, i.e., $\mathcal{S} = \{x : (x_1 - x_1^*)^2 + (x_2 - x_2^*)^2 - r^2 \geq 0\}$. The CBF proposed in [174] is used. The desired control input is a LQR controller linearized about the hover state. The observer is a DEKF adapted from [137]:[12] Defining constant matrices $D_1 = g_d(x)$ and $D_2 = c_d(x)$, the observer is

$$\begin{aligned}
\dot{\hat{x}} &= f(\hat{x}) + g(\hat{x})u + PC^T R^{-1}(y - c(\hat{x})) \\
\dot{P} &= PA^T + AP - PC^T R^{-1} CP + Q + 2\theta P \\
\dot{V} &= -2\theta V + 2\sqrt{V}\left(\left\|D_1^T P^{-1/2}\right\| \bar{d} + \left\|(LD_2)^T P^{-1/2}\right\| \bar{v}\right)
\end{aligned}$$

where $\theta \geq 0$ is a design parameter, $A = \frac{\partial}{\partial \hat{x}}(f(\hat{x}) + g(\hat{x})u)$, $C = \frac{\partial c}{\partial x}(\hat{x})$. In the standard form of EKFs [105, Sec 5.3], the disturbances are assumed to be Weiner processes and Q, R represent the covariances of the $d(t)$ and $v(t)$. However in the Deterministic EKF, we assume $d(t), v(t)$ are bounded, and thus $Q \in \mathbb{S}^n_{++}, R \in \mathbb{S}^{n_y}_{++}$ can be freely chosen. Assuming there exist positive constants p_1, p_2 such that $p_1 I \leq P(t) \leq p_2 I \ \forall t \in \mathcal{I}$, (see [91, Sec 11.2]), this observer is a Bounded-Error observer, and satisfies (2.54) with $\mathcal{D}(\hat{x}_0) = \mathcal{P}(0, \hat{x}_0)$, and $\mathcal{P}(t, \hat{x}) = \{x : (x - \hat{x})^T P(t)^{-1}(x - \hat{x}) \leq V(t)\}$.

The method in Approach 2 is used to synthesize the interconnected observer-controller. Specifically, the functions a, b_i^-, and b_i^+ were determined using Lipschitz bounds, and the QP (2.56) is used to determine the control input.

Figure 2.6 compares the trajectory of the planar quadrotor using the controller proposed in [174] (baseline case) to the proposed controller of Approach 2. In the baseline case, since the state estimate is used in lieu of the true state, safety is violated. By accounting for the state estimation uncertainty, the proposed controller avoids the obstacle.

[12] In [137], only the undisturbed case is demonstrated. The extension to include bounded disturbances can be derived using the same techniques as in the original paper. The additional terms due to the disturbances are bounded using [91, Eq. B4].

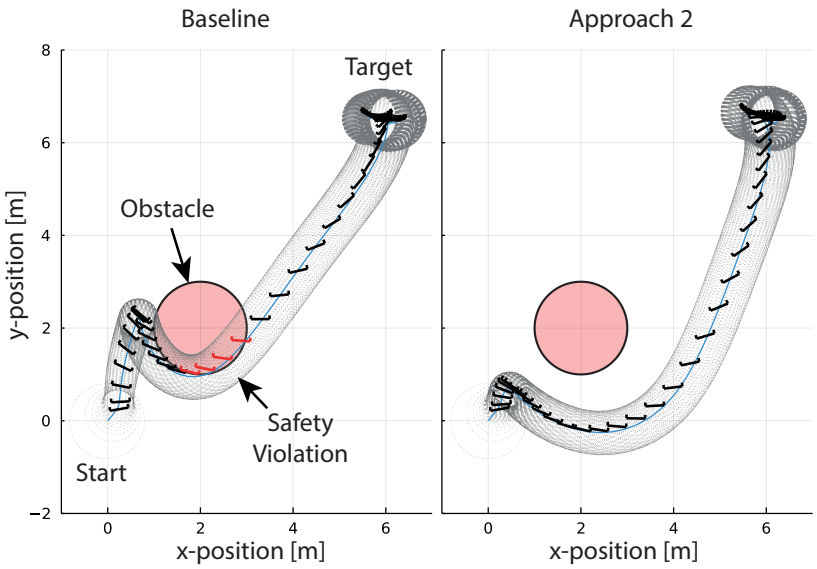

Figure 2.6: Simulation Results for the Planar Quadrotor. The objective is to fly the quadrotor from the starting state to the target position while avoiding the circular obstacle region. The blue lines indicate the path of the state estimate and grey lines the the projection of $\mathcal{P}(t, \hat{x})$ on the x-y plane. The icons show the quadrotor's true position every 0.2 s and is colored red while violating safety. (a) uses the baseline CBF controller, and (b) uses Approach 2.

Experiments: 3D Quadrotor

For our experiments, we use the Crazyflie 2.0 quadrotor, using the on-board IMU and barometer sensors and an external Vicon motion capture system. The objective was to fly in a figure of eight trajectory, but to not crash into a physical barrier placed at $x = 0.5$ meters. State was estimated using an EKF [123], assuming the true state lies within the 99.8% confidence interval of the EKF. To design the controller, first $\pi_{des}(t, \hat{x})$ is computed using an LQR controller, which computes desired accelerations wrt to an inertial frame to track the desired trajectory. This command is filtered using a safety critical QP, either the baseline CBF-QP (Figure 2.7a) or the proposed QP using Approach 2 (2.56) (Figure 2.7c). Finally, the internal algorithm of the Crazyflie (based on [116]) is used to map the output of the QP to motor PWM signals. The magnitude of the disturbances was estimated by collecting experimental data when the quadrotor was commanded to hover. The trajectories from the two flight controllers are compared in Figure 2.7. In the baseline controller, the quadrotor slows down as it approaches the barrier, but still crashes into barrier. In the proposed controller, the quadrotor remains safe, Figure 2.7e.

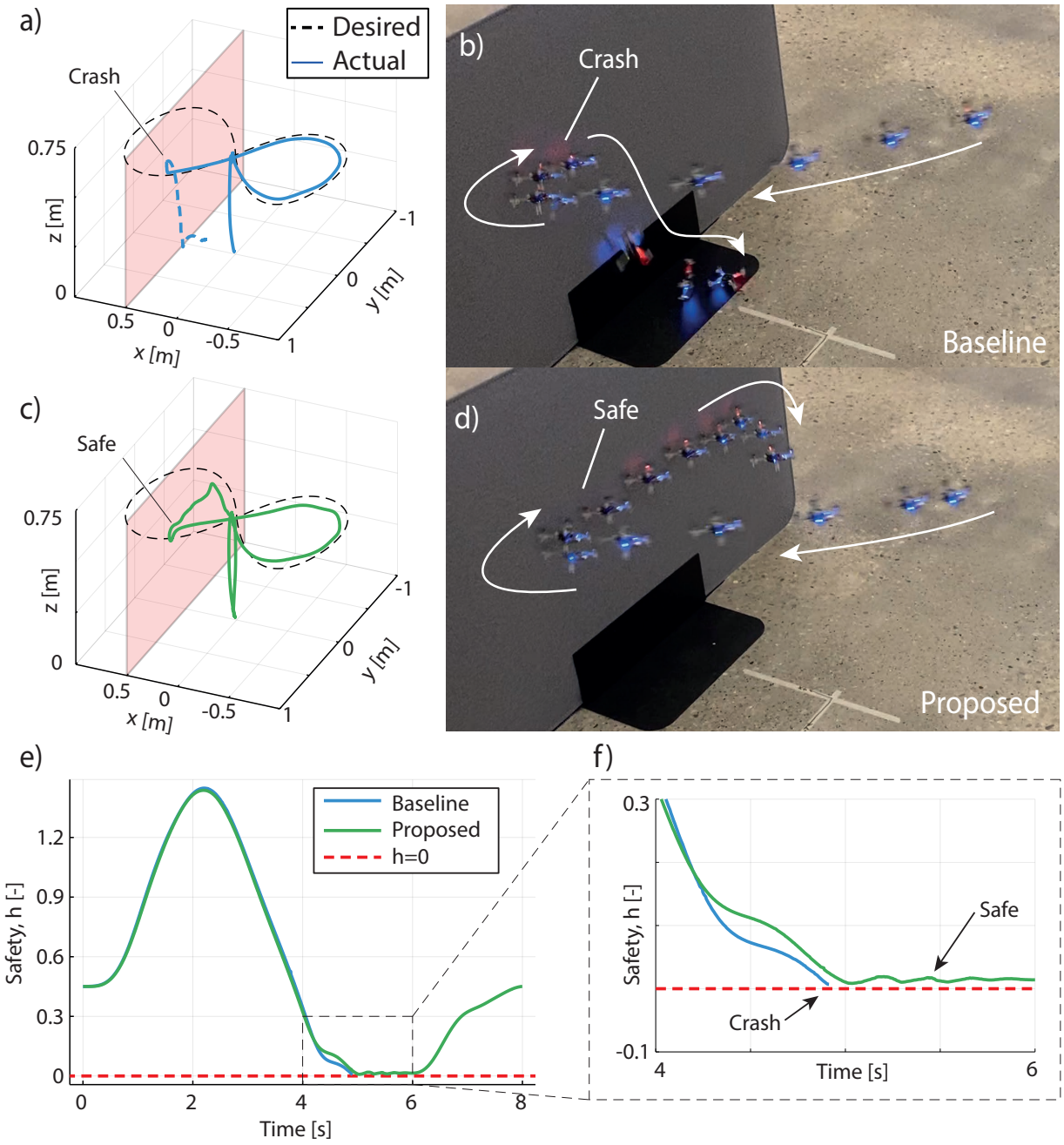

Figure 2.7: Experimental results. The quadrotor is commanded to track a figure-of-eight trajectory, while avoiding the physical barrier at $x = 0.5$ m. Ground truth trajectories are plotted in (a, c) for the baseline CBF and proposed controllers respectively. Snapshots from the experiment are show in (b, d). (e, f) Plots of the safety value, h over time for both trajectories.

2.3.4 Conclusion

We have developed two methods to synthesize observer-controllers that are robust to bounded disturbances on system dynamics and measurements, and maintain safety in the presence of imperfect information. We have demonstrated the efficacy of these methods in simulation and experiments. Future work will investigate methods to learn the disturbance, such that the controller can adaptively tune itself to achieve better performance, and to extend the work to handle probabilistic guarantees of safety when the system is subject to stochastic disturbances instead of bounded disturbances.

CHAPTER 3

Safety-Critical Planning

In the previous chapter we focused on the design of the controller as it pertains to safety-critical design. In both ICCBFs and Observer-Robust CBFs, we saw the controller can filter the inputs π_{des}, a desired control input, to ensure that a safe control signal is sent to the robot's actuators.

Although few constraints were placed on this signal π_{des}, seemingly the system can achieve safety. Why then did we suggest in Chapter 1 that there was a flow of constraints from the controller upstream towards the planner? In a nutshell, the problem is that finding a CBF is a difficult task - it depends on the system dynamics and the safe set definition, and thus for each new scenario one must attempt to find a CBF for their problem. Furthermore, when there are multiple constraints as is often the case in practical robotic systems, many of the results of CBFs become inapplicable.

We will propose two different methods to solve this problem. Both adopt the position that to guarantee safety, some of the safety critical constraints must be satisfied at the planning level instead of at the control level. This allows the autonomy stack to avoid entering situations where it will eventually no longer be possible to act safely.

Both methods rely on different properties of the system. The first uses a property called differential flatness, which allows one to find an equivalent linear system of a nonlinear system. Then parts of the planning and control can take place in the lifted linear space, and the results mapped back to the nonlinear space. Many practical robot systems (including all Euler-Lagrange systems) posses the differential flatness property, but the algebra is somewhat involved, and some constraints can be difficult to map from the nonlinear space to the linear space. Nonetheless, we demonstrate through examples how the method can be used.

In the second method, we rely on the existence of backup controllers. This approach is remarkably simple, and since almost all robotic systems have a failsafe method, this approach can be widely applied. In fact, multiple members of our lab have adopted this as the underlying principle that enables their more complicated algorithms and applications.

3.1 Multirate Planner-Controllers using Differential Flatness

Control of nonlinear systems for navigating a constrained environment is a common problem in safety-critical robotics. Despite the extensive work on planning and control methods, the real-time deployment of controllers that provide guarantees of safety poses challenges either due to the computational complexity of nonlinear and nonconvex optimization, or due to the curse of dimensionality in search based approaches [25].

Recently, multirate controllers have shown promising results for combining planning and tracking [20, 66, 68, 74, 78, 93, 94, 142, 151, 168, 180, 181]. Safety can be guaranteed using low-level filters that compute the closest safe control action to a desired command using control barrier functions [20, 74, 168]. The tracking error and control policy can be computed using Hamilton-Jacobi (HJ) reachability analysis [78] or sum-of-squares programming [151, 180]. Nonlinear tube MPC approaches have been developed to bridge high-level planning with low-level control [66, 68, 93, 94, 142, 181]. However the nonlinearity in the MPC poses a challenge in identifying suitable barrier functions in the tracking layer. In [61], a tube-based planning approach with safety guarantees was developed. However since the size of the tube increases over the planning horizon, the approach is conservative and cannot be used for in recursive planning methods like MPC.

Many dynamical systems, including unicycles, quadrotors, inverted pendulums, and induction motors, possess a useful property known as differential flatness (see [114, Ch. 7] for a catalog of flat systems). Several studies have identified that such systems possess useful properties for planning [62, 114, 116, 124, 168]. For instance in [116], quadrotor trajectories from initial to target flat states are represented as polynomials whose coefficients are determined by solving a linear system. However, the method does not provide a principled way of incorporating disturbances or safety constraints.

In this work, we propose a novel multirate controller that leverages properties of differentially-flat systems. These properties allow a constructive means of designing both the planner, a linear MPC, and the tracker, an Input-to-State Stable (ISS) feedback controller. We provide formal guarantees of safety for the continuous-time (low-level) system, and recursive feasibility for the MPC planner. We experimentally demonstrate our framework on a ground rover and a quadrupedal robot that can be modeled as unicycles.

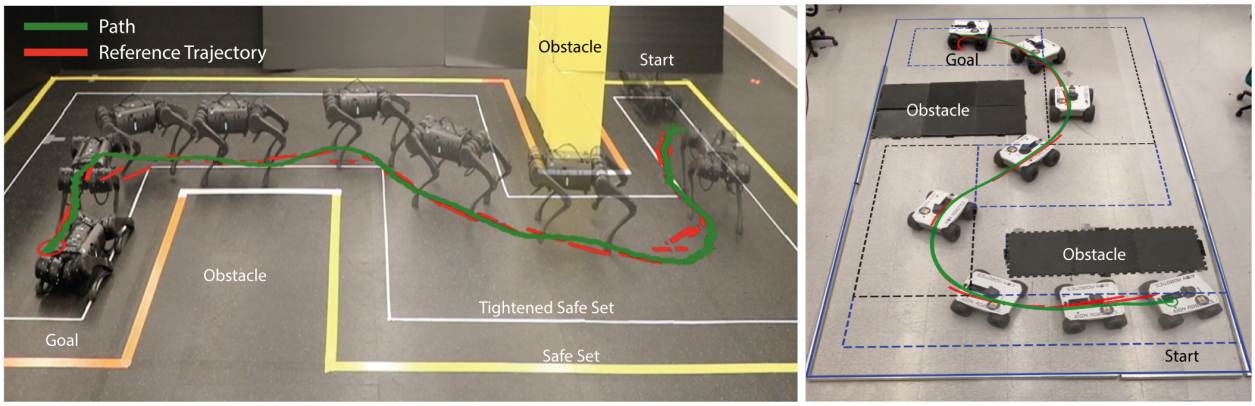

Figure 3.1: Snapshots of a quadrupedal robot (left) and ground rover (right) navigating safely from start to goal positions around two rectangular obstacles. The safe set (thick outside line) and the tightened set (thin/dashed lines) are shown. The reference trajectory (red) is solved online using Model Predictive Control, and must lie inside the tightened safe set. A tracking controller ensures the maximum deviation from this reference trajectory is smaller than the tightening. Thus the true path (green) remains within the safe set. Video: https://github.com/dev10110/Multirate-Controllers-for-Differentially-Flat-Systems.

3.1.1 Preliminaries

3.1.1.1 Differentially-Flat Systems

Consider a control-affine nonlinear system

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

where $x \in \mathcal{X} \subset \mathbb{R}^n$ is the state, and $u \in \mathbb{R}^m$ is the control input. The functions $f : \mathcal{X} \to \mathbb{R}^n$ and $g : \mathcal{X} \to \mathbb{R}^{n\times m}$ are smooth and satisfy $f(0) = 0$. Differential flatness of such systems is often defined in terms of flat outputs (as in [116]) or using differential geometry (e.g. [63, 114]). Following [102], here we define it in terms of endogenous dynamic feedback.[1]

Definition 3.1. *The control system* (3.1) *is* ***differentially flat*** *over a domain* $\mathcal{M} \subset \mathcal{X} \times \mathbb{R}^q$ *if (I) there exists an* ***endogenous dynamic feedback***

$$\dot{y} = a(x, y, v) \tag{3.2}$$
$$u = b(x, y, v), \tag{3.3}$$

where $y \in \mathbb{R}^q$ *is an additional state (referred to as the* ***dynamic extension****) and* $v \in \mathbb{R}^m$ *is a different control input (referred to as the* ***flat control input****, and (II) there exists a*

[1] This is possible since [102, Thm. 3] shows that a system is differentially flat if and only if it admits an endogenous dynamic feedback. See [103, Sec 5.3.6] for an explanation of the term *endogenous*.

diffeomorphism $\Xi : \mathcal{M} \to \mathcal{N}$*, where* $\mathcal{N} \subset \mathbb{R}^n \times \mathbb{R}^q$ *is a domain,*

$$\xi = \Xi(x, y), \tag{3.4}$$

which maps the nonlinear states (x, y) *to the* ***flat state*** $\xi \in \mathbb{R}^{n+q}$ *such that the dynamics of* ξ *are linear, time-invariant*

$$\dot{\xi} = \frac{\partial \Xi}{\partial x}\dot{x} + \frac{\partial \Xi}{\partial y}\dot{y} = A\xi + Bv, \tag{3.5}$$

where A, B *are constant matrices of appropriate size.*

Remark 3.1. *When a system is differentially flat, the flat system is linear, time-invariant, and controllable [63].*

Remark 3.2. *If* $q = 0$ *(the dimension of* y*), the dynamic feedback is equivalent to a full-state feedback, and thus full-state feedback linearizable systems are a subset of differentially-flat systems (subject to smoothness requirements) [135].*

Example 3.1 (Unicycle: Differential flatness). The unicycle is a differentially-flat system with nonlinear dynamics

$$\dot{x}_1 = u_1 \cos x_3, \quad \dot{x}_2 = u_2 \sin x_3, \quad \dot{x}_3 = u_2,$$

where (x_1, x_2) is the position, x_3 is the heading angle, and the control inputs are linear and angular velocities, u_1, u_2. The dynamic extension is $y = [\dot{x}_1, \dot{x}_2]^T$, and flat state is $\xi \in \mathbb{R}^4$. Then $\Xi(x, y) = [x_1, x_2, y_1, y_2]^T$. The flat system dynamics are

$$\dot{\xi}_1 = \xi_3, \quad \dot{\xi}_2 = \xi_4, \quad \dot{\xi}_3 = v_1, \quad \dot{\xi}_4 = v_2,$$

where $v \in \mathbb{R}^2$ is the flat control input. The nonlinear state and control input can be determined from ξ and v as

$$\begin{aligned} x_1 = \xi_1, \quad x_2 = \xi_2, \quad x_3 = \arctan\left(\xi_4/\xi_3\right), \\ u_1 = \sqrt{\xi_3^2 + \xi_4^2}, \quad u_2 = \frac{-\xi_4 v_1 + \xi_3 v_2}{\xi_3^2 + \xi_4^2}. \end{aligned}$$

Notice that Ξ^{-1} has a singularity at $\dot{x}_1 = \dot{x}_2 = 0$, and thus is excluded from the domain $\mathcal{M}$. See Example 2 and [114, Sec. 2.5] on methods to handle singularities.

3.1.1.2 Input-To-State Stability

Assume that a bounded additive disturbance[2] $w : \mathbb{R}_{>0} \to \mathbb{R}^n$, $w \in \mathbb{L}_\infty^n$ is introduced to the system (3.1), and that a feedback controller $\pi : \mathcal{X} \to \mathbb{R}^m$, $\pi(0) = 0$ has been designed. Then the closed-loop system dynamics read:

$$\dot{x} = f(x) + g(x)\pi(x) + w(t), \tag{3.6}$$

where $\|w(t)\|_\infty \triangleq \sup_{t\geq 0}\{|w(t)|\} \leq \bar{w}$ for some $\bar{w} < \infty$.

Definition 3.2. *A controller $\pi : \mathcal{X} \to \mathbb{R}^m$, $\pi(0) = 0$ and system (3.6) are **input-to-state stabilizing** and **input-to-state stable**, respectively, wrt. w, if there $\exists \beta \in \mathcal{KL}$, $\iota \in \mathcal{K}_\infty$ s.t.*

$$\|x(t, x_0, w)\| \leq \beta(\|x_0\|, t) + \iota(\|w\|_\infty) \tag{3.7}$$

for all $x_0 \in \mathcal{X}, w \in \mathbb{L}_\infty^n$ and $t \geq 0$.

Definition 3.3. *A continuously differentiable positive definite function $V : \mathbb{R}^n \to \mathbb{R}_{>0}$ is an **input-to-state stabilizing control Lyapunov function (ISS-CLF)** with respect to w, if there exists functions $\alpha, \iota \in \mathcal{K}_\infty$ such that $\forall x \in \mathcal{X}$ and $w \in \mathbb{L}_\infty^n$,*

$$\inf_{u\in\mathbb{R}^m} \left[L_f V(x) + L_g V(x)u + \frac{\partial V}{\partial x} w \right] \leq -\alpha(\|x\|) + \iota(\|w\|_\infty). \tag{3.8}$$

Definition 3.4. *For the system (3.6), a set $\mathcal{D} \subset \mathbb{R}^n$ is **robustly control invariant**, with respect to disturbances w bounded by $\bar{w}$, if there exists a feedback controller $\pi(x) \in \mathbb{R}^m$ such that $x(t_0) \in \mathcal{D} \implies x(t) \in \mathcal{D}$ for all $t \geq t_0$ and for all $w \in \mathbb{L}_\infty^m$ where $\|w(t)\|_\infty \leq \bar{w}$.*

3.1.2 Controller Construction

Our multirate controller consists of two stages: A high-level planner, in the form of a linear MPC, and a low-level tracker, in the form a feedback controller. In designing the low-level tracker, we define explicitly a set $\mathcal{D}$, which is the set of possible tracking errors between the reference and current state. In the high-level, we shrink the safe set by $\mathcal{D}$ and require the MPC to generate reference trajectories that lie within the tightened safe set. As a consequence, the system's trajectory will lie in the safe set, despite the disturbances.

[2]ISS is typically defined for *matched disturbances* $w = g(x)d$ [156]. In this paper, we consider the *unmatched case* as necessitated by the coordinate transformations resulting from differential flatness.

3.1.2.1 High-Level Planning

At the high-level, we solve a Finite Time Optimal Control Problem (FTOCP) every T seconds. The prediction horizon is N steps, i.e., NT seconds. $\xi(t)$ denotes the continuous-time flat state at time t. $z_{i|k}$ denotes the predicted flat state at time $t = iT$, when the FTOCP is solved at time $t = kT$. We minimise a cost function over the sequence of flat states $\boldsymbol{z}_k = [z_{k|k}, z_{k+1|k}, ..., z_{k+N|k}]$ and flat control inputs $\boldsymbol{v}_k = [v_{k|k}, v_{k+1|k}, ..., v_{k+N-1|k}]$ subject to (A) the given dynamics, (B) initial and final constraints and (C) safety constraints. The goal state is $\xi_g \in \mathbb{R}^{n+q}$, a flat state corresponding to a target state x_g of the nonlinear system, i.e., $\xi(t) = \xi_g \implies x(t) = x_g$, where $(x(t), y(t)) = \Xi^{-1}(\xi(t))$.

The FTOCP problem is the following optimization problem:

$$
\begin{aligned}
J^*(\xi(kT)) = \min_{\boldsymbol{z},\boldsymbol{v}} \quad & \sum_{i=k}^{k+N-1} l(z_{i|k}, v_{i|k}) && \text{(3.9a)}\\
\text{s.t.} \quad & z_{i+1|k} = A_d z_{i|k} + B_d v_{i|k}, && \text{(3.9b)}\\
& z_{k|k} - \xi(kT) \in \mathcal{D}, && \text{(3.9c)}\\
& z_{k+N|k} = \xi_g, && \text{(3.9d)}\\
& (z_{i|k}, v_{i|k}) \in \mathcal{H} && \text{(3.9e)}\\
& \quad \forall i \in \{k, ..., k+N-1\}.
\end{aligned}
$$

Cost Function, (3.9a): $l(\xi, v)$ is the stage cost of action v from a state ξ. We assume l is convex in both arguments, is positive definite about $(\xi_g, 0)$, and is radially unbounded.

Dynamics, (3.9b): Under a zero-order hold, (3.9b) is the exact discretisation of the flat system (3.5) i.e., $A_d = \exp(AT), \quad B_d = \int_0^T \exp(A\tau) B d\tau$.

Initial Condition, (3.9c)***:*** The initial state $z_{k|k}$, will be chosen by the FTOCP to be in the neighborhood of the flat state:

$$
(z_{k|k} - \xi(kT)) \in \mathcal{D}, \tag{3.10}
$$

where $\mathcal{D}$ is an ellipsoid, and will be defined later, in (3.23). Intuitively, $\mathcal{D}$ represents the maximum tracking error between the reference state $\xi_{\text{ref}}(t)$, and the current state $\xi(t)$.

Final Condition, (3.9d)***:*** The final state $z_{k+N|k}$ must be the goal state ξ_g, which is assumed to be a safe unforced equilibrium point for the system, i.e., $z = A_d z$.

Safety, (3.9e): Let $\mathcal{S} \subset \mathbb{R}^n$ be the set of safe states for the nonlinear system (3.1). Then the set of safe flat states is $\mathcal{C} \subset \mathbb{R}^{(n+q)}$, where $\mathcal{C} = \{\xi \in \mathcal{N} : x \in \mathcal{S}, (x, y) = \Xi^{-1}(\xi) \in \mathcal{M}\}$. Next, we define the set $\mathcal{H}$ such that a flat state-input pair chosen in $\mathcal{H}$ ensures that the

intersample trajectory remains within a subset of $\mathcal{C}$. Mathematically, $\mathcal{H} \subset \mathbb{R}^{n+q+m}$ is s.t.:

$$(z_{i|k}, v_{i|k}) \in \mathcal{H} \Rightarrow \xi_{\text{ref}}(t) \in \mathcal{C} \ominus \mathcal{D}, \forall t \in [iT, (i+1)T) \tag{3.11}$$

where $\xi_{\text{ref}}(t)$ is the solution to $\dot{\xi}_{\text{ref}} = A\xi_{\text{ref}} + Bv_{i|k}$ from the initial condition $\xi_{\text{ref}}(iT) = z_{i|k}$. While computing $\mathcal{C}$ and $\mathcal{H}$ could be challenging, in many physical systems the safe set can be described in terms of the flat states. In these cases, constructing $\mathcal{C}$ can be straightforward. To compute $\mathcal{H}$, reachability-based methods for linear systems can be used, e.g., [172]. The example below demonstrates an approach based on control barrier functions, e.g., [36, 152].

The final result, by solving the FTOCP (3.9), is a sequence of flat states $\boldsymbol{z}_k$ and flat control inputs $\boldsymbol{v}_k$. The reference trajectory and input for the continuous-time flat system are defined for all $t \in [0, NT)$ using index $i \triangleq \text{floor}(t/T)$ as

$$\xi_{\text{ref}}(t) = e^{A(t-iT)} z_{i|k} + \left(\int_0^{t-iT} e^{A\tau} d\tau \right) B v_{i|k}, \tag{3.12a}$$

$$v_{\text{ref}}(t) = v_{i|k}, \tag{3.12b}$$

Remark 3.3. *The FTOCP problem is a convex problem. Constraint* (3.9c) *is a quadratic constraint, and the rest are linear constraints. As such, it is a second order cone program (SOCP), and can be solved efficiently using interior point methods or specialised solvers for MPC problems [35, Ch. 4,11][17].*

Example 3.2 (Unicycle: High Level Planner[3]). Here, we provide details on computing $\mathcal{H}$. Figure 3.1 shows the safe set $\mathcal{S}$. Since the first two components of ξ and x are equal, $\mathcal{C} = \{\xi : [\xi_1, \xi_2] \in \mathcal{S}\}$. Since the tracking error lies in an ellipsoid $\mathcal{D}$, we shrink the safe set by the diameter of $\mathcal{D}$, and refer to this set as the *tightened safe set*. This region is partitioned into 5 regions with overlapping boundaries, indicated by dashed rectangles. To find $\mathcal{H}$, we constrain the trajectory due to a state-input pair $(z_{i|k}, v_{i|k})$ to lie within a single rectangular region for the next T seconds. To do this, four sampled control barrier functions [36] are defined, one for each edge of the rectangle: each edge defines a half space of the form $a^T\xi \leq b$. The barrier function is $h(\xi) = a^T\xi - b$, such that the safe set with respect to this edge is $\{\xi : h(\xi) \leq 0\}$. A sufficient condition for $h(\xi(t)) \leq 0$ for all $t \in [(i+k)T, (i+k+1)T)$ is

$$h_0 \leq 0 \text{ and } h_0 + \dot{h}_0 T + 1/2 \max\{0, \ddot{h}_0\} T^2 \leq 0,$$

where $h_0 = a^T z_{i|k} - b$ and $\dot{h}_0, \ddot{h}_0$ are the first and second derivatives of h at $z_{i|k}, v_{i|k}$. This is

[3]The code for the FTOCP is at https://github.com/dev10110/Multirate-Controllers-for-Differentially-Flat-Systems

a linear constraint in both $z_{i|k}, v_{i|k}$, but requires integers to index the rectangles. Thus the FTOCP is a Mixed-Integer SOCP, which is solved online.

With regards to singularities, our method guarantees safety at all states where Ξ is non-singular, (i.e., in the domain $\mathcal{M}$ of Ξ). For the unicycle, since the singular points correspond to states where the unicycle is at rest (see Example 1), these states are intrinsically safe. As such, in our planning problem, we can specify stopping at a desired location as a suitable target state. In general, independent analysis is needed to ensure the singular points are intrinsically safe.

3.1.2.2 Low-Level Tracking

In this section, we construct a tracking controller. Consider a potentially time-varying, bounded matched disturbance $d \in \mathbb{L}_\infty^m$, $\|d\|_\infty \leq \bar{d} < \infty$. The disturbed system is

$$\dot{x} = f(x) + g(x)u + g(x)d(t). \tag{3.13}$$

Transforming the disturbed system (3.13) to the flat space,

$$\begin{aligned} \dot{\xi} &= \frac{\partial \Xi}{\partial x}\dot{x} + \frac{\partial \Xi}{\partial y}\dot{y} && (3.14)\\ &= \frac{\partial \Xi}{\partial x}\big(f(x) + g(x)u + g(x)d(t)\big) + \frac{\partial \Xi}{\partial y}\dot{y} \\ &= A\xi + Bv + \frac{\partial \Xi}{\partial x}g(x)d(t), && (3.15) \end{aligned}$$

where A, B are as in (3.5), and Ξ is defined in (3.4). Let $\xi_e \triangleq \xi - \xi_{\text{ref}}$, $v_e \triangleq v - v_{\text{ref}}$ and $w \triangleq \frac{\partial \Xi}{\partial x}g(x)d(t)$. Then

$$\dot{\xi}_e = A\xi_e + Bv_e + w, \tag{3.16}$$

Assume the disturbance w is bounded, $\|w\|_\infty \leq \bar{w}$. Define the tracking feedback controller $\pi_e : \mathbb{R}^{n+q} \to \mathbb{R}^m$, and Lyapunov function

$$\pi_e(\xi_e) = -\frac{1}{2}R^{-1}BP\xi_e \tag{3.17}$$

$$V(\xi_e) = \frac{1}{2}\xi_e^T P \xi_e, \tag{3.18}$$

where $P \in \mathbb{R}^{(n+q)\times(n+q)}$ is a symmetric positive definite matrix satisfying a modified Riccati equation

$$PA + A^T P - PBR^{-1}B^T P = -Q - PP/\gamma^2, \tag{3.19}$$

where $Q \in \mathbb{R}^{(n+q)\times(n+q)}, R \in \mathbb{R}^{m\times m}$ are user-specified symmetric positive definite matrices. The parameter $\gamma > 0$ must be chosen such that P exists. Since (A, B) is controllable, P exists for a sufficiently large γ [97].

Lemma 3.1. *The function V (3.18), is an ISS-CLF for the flat error system (3.16), wrt. the bounded disturbance w, $\|w\|_\infty \leq \bar{w}$, under the feedback law (3.17), with α and ι defined as*

$$\alpha(\|\xi_e\|) = \frac{1}{2}\lambda_{min}(Q)\|\xi_e\|^2, \quad \iota(\bar{w}) = \frac{1}{2}\gamma^2\bar{w}^2. \tag{3.20}$$

Proof. The time derivative of V along the closed-loop trajectories of system (3.16) and feedback law (3.17), is

$$\begin{aligned}\frac{d}{dt}V &= \frac{1}{2}\xi_e^T\left(-Q - \frac{1}{\gamma^2}PP\right)\xi_e + \xi_e^T P w \\ &\leq -\frac{1}{2}\xi_e^T Q\xi_e - \frac{\|P\xi_e\|^2}{2\gamma^2} + \|P\xi_e\|\,\|w\|.\end{aligned} \tag{3.21}$$

Adding and subtracting $\frac{1}{2}\gamma^2\|w\|^2$ and factorizing yields

$$\frac{d}{dt}V \leq -\frac{1}{2}\xi_e^T Q\xi_e - \frac{1}{2}\left(\frac{\|P\xi_e\|}{\gamma} - \gamma\|w\|\right)^2 + \frac{1}{2}\gamma^2\|w\|^2. \tag{3.22}$$

Thus V is a ISS-CLF (3.8), with α, ι defined in (3.20). □

Lemma 3.2. *For the closed-loop flat error system (3.16), under the feedback law (3.17), bounded disturbances w, $\|w\|_\infty \leq \bar{w}$, and V as in (3.18), the sub-level set*

$$\mathcal{D} = \{\xi_e : V(\xi_e) \leq V_{max}\}, \tag{3.23}$$

is robustly control invariant, where

$$V_{max} = \frac{1}{2}\gamma^2\frac{\lambda_{max}(P)}{\lambda_{min}(Q)}\bar{w}^2. \tag{3.24}$$

Proof. Using $\dot{V} \leq -\alpha(\|\xi_e\|) + \iota(\bar{w})$, we have $\alpha(\|\xi_e\|) \geq \iota(\bar{w}) \implies \dot{V} \leq 0$, and therefore

$$\|\xi_e\|^2 \geq \gamma^2 \bar{w}^2 / \lambda_{min}(Q) \implies \dot{V} \leq 0. \tag{3.25}$$

Similarly, since $V(\xi_e) \leq \frac{1}{2}\lambda_{max}(P)\|\xi_e\|^2$, we have $V(\xi_e) \geq V_{max} \implies$ (3.25) and thus $\dot{V} \leq 0$, completing the proof. □

Example 3.3 (Unicycle: Low Level Tracking). For the unicycle, we have $\bar{w} = \bar{d}$, since

$$\begin{aligned}\bar{w} \triangleq \left\| \frac{\partial \Xi}{\partial x} g(x) d(t) \right\|_\infty &= \left\| \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \cos x_3 & 0 \\ \sin x_3 & 0 \\ 0 & 1 \end{bmatrix} d(t) \right\|_\infty \\ &= \|d(t)\|_\infty \triangleq \bar{d}.\end{aligned}$$

The Riccati equation with $Q = I_4, R = I_2, \gamma = 2$, defines P in (3.19), the controller in (3.17) and $\mathcal{D}$ in (3.23), with $V_{max} = 9.66\bar{w}^2$.

3.1.3 Main Result

To summarise, the multirate controller is as follows. At the high-level, every T seconds, the FTOCP (3.9) is solved. Using (3.12), the continuous-time reference trajectory $\xi_{\text{ref}}(t), v_{\text{ref}}(t)$ is determined. At the low-level, the flat error state ξ_e, and the flat input $v = v_{\text{ref}} + \pi_e(\xi_e)$ are computed using (3.17). Finally, the nonlinear control input is computed using (3.3), $u = b(x, y, v)$, where $(x, y) = \Xi^{-1}(\xi)$.

We make the following assumptions and prove two lemmas before presenting the main result.

Assumption 3.1. *The unmatched disturbances w in the flat system* (3.16) *are bounded,* $\|w\|_\infty \leq \bar{w}$.

Assumption 3.2. *The state of the system (nonlinear and flat state) is perfectly measured at all times.*

Assumption 3.3. *At the initial time t_0,* (3.9) *is feasible.*[4]

Remark 3.4. *In the case of compact safe sets $\mathcal{S}$, as in many practical applications, Assumption 3.1 can be verified, since $\partial\Xi/\partial x$ and $g(x)$ can be bounded, for instance using Lipschitz constants. See Example 3 for the closed form expression for $\bar{w}$ for the unicycle system.*

[4]Since FTOCP (3.9) is a convex program, the set of feasible initial conditions can be computed explicitly [34].

Lemma 3.3. *Consider a reference trajectory* $(\xi_{\text{ref}}(t), v_{\text{ref}}(t))$ *defined over the interval* $t \in [t_0, t_1]$ *that satisfies the flat dynamics* (3.5)*. Under Assumption 3.1, if at* t_0*, the flat error* $\xi(t_0) - \xi_{\text{ref}}(t_0) \in \mathcal{D}$*, then the flat feedback control law*

$$\pi(t, \xi) = v_{\text{ref}}(t) + \pi_e(\xi(t) - \xi_{\text{ref}}(t)), \tag{3.26}$$

where π_e*,* (3.17)*, ensures tracking error remains within* $\mathcal{D}$*:*

$$\xi(t) - \xi_{\text{ref}}(t) \in \mathcal{D}, \quad \forall t \in [t_0, t_1]. \tag{3.27}$$

Proof. The proof is immediate since $\mathcal{D}$ is a robust control invariant set for the flat error system (Lemma 3.2). □

Lemma 3.4. *Under Assumptions 3.1-3.3, and if the flat system is controlled using the tracking controller* (3.26)*, the FTOCP* (3.9)*, is recursively feasible.*

Proof. Let the solution to (3.9) at the initial time t_0 be $\boldsymbol{z}_0, \boldsymbol{v}_0$. We show that

$$\begin{aligned} \boldsymbol{z}_1 &= [z_{1|0}, z_{2|0}, ..., z_{N-1|0}, z_{N|0}, \xi_g], \\ \boldsymbol{v}_1 &= [v_{1|0}, v_{2|0}, ..., v_{N-1|0}, 0], \end{aligned} \tag{3.28}$$

is a feasible solution at the subsequent step $t = t_0 + T$. Since $z_{N|0} = \xi_g$ and $A_d \xi_g = \xi_g$, it is immediate that $\boldsymbol{z}_1, \boldsymbol{v}_1$ satisfies dynamics (3.9b), final condition (3.9d) and safety requirement (3.9e). Constraint (3.9c) remains to be shown. From the solution $\boldsymbol{z}_0, \boldsymbol{v}_0$, the reference trajectory-input pair $\xi_{\text{ref}}(t), v_{\text{ref}}(t)$ for $t \in [t_0, t_0 + NT]$ is computed (3.12). Since $z_{0|0} = \xi_{\text{ref}}(t_0)$, $\xi(t_0) - \xi_{\text{ref}}(t_0) \in \mathcal{D}$. By Lemma 3.3, the low-level tracking controller guarantees that at the next step

$$\xi(t_0 + T) - z_{1|0} \in \mathcal{D}, \tag{3.29}$$

since $\xi_{\text{ref}}(t_0 + T) = z_{1|0}$. Thus, the initial constraint (3.9c) is also satisfied, completing the proof. □

Theorem 3.5 (Main Result)**.** *Consider the nonlinear system* (3.13)*, subject to bounded, matched disturbances* $d(t)$*,* $\|d\|_\infty \leq \bar{d}$*. Under Assumptions 3.1- 3.3, the proposed controller* (3.9)*,* (3.12)*,* (3.26)*,* (3.3) *is recursively feasible. Furthermore, the closed-loop system trajectories satisfy safety:* $x(t) \in \mathcal{S}$ $\forall t \geq t_0$*, and as* $k \to \infty$*,* $x(kT) \in \{x : \exists y \text{ s.t. } \Xi(x, y) - \xi_g \in \mathcal{D}\}$*, i.e., a neighborhood of* x_g *is reached.*

Proof. First, we prove the recursive feasibility of the FTOCP: Since the low-level controller dictates u such that the conditions of Lemma 3.4 are met, recursive feasibility is maintained.

Second, we prove safety is maintained: By definition of $\mathcal{H}$ (3.11), the FTOCP waypoints $\boldsymbol{z}$ and control inputs $\boldsymbol{v}$ are such that the reference trajectory $\xi_{\text{ref}}(t)$ remains within the set $\mathcal{C} \ominus \mathcal{D}$. The low-level controller guarantees that the tracking error ξ_e remains in $\mathcal{D}$, i.e., $\xi(t) \in \mathcal{C}$ for all $t \geq t_0$. Since $\xi \in \mathcal{C} \implies x \in \mathcal{S}$, the nonlinear state remains safe.

Finally, we prove convergence to goal state: Let $J^*(\xi(kT))$ be optimal cost at $t = kT$. Cost associated with the feasible solution (3.28) is $\bar{J}(\xi((k+1)T)) = J^*(\xi(kT)) - l(z_{0|0}, v_{0|0})$. Therefore, optimal cost at $t = (k+1)T$ satisfies

$$\begin{aligned} J^*(\xi(kT)) &\geq \bar{J}(\xi((k+1)T)) = J^*(\xi(kT)) - l(z_{0|0}, v_{0|0}) \\ &\geq J^*(\xi((k+1)T)). \end{aligned}$$

From the above equation and the positive definiteness of the stage cost l, the optimal cost $J^*(\xi(kT))$ is a Lyapunov function for the closed loop system (3.9b), (3.12b). The result from Lemma 3.2 implies that $\lim_{k\to\infty} \xi(kT) \in \{\xi_g\} \oplus \mathcal{D}$. Together with the invertibility of Ξ, this implies $x(t)$ reaches the neighborhood of x_g defined in the theorem. □

Discussion. The proposed controller differs from other planner-tracker controllers since the two levels are coupled through $\mathcal{D}$, enabling formal guarantees on feasibility and safety. Furthermore, the controller is constructive given the diffeomorphism Ξ and a few parameters.[5] The controller also provides greater flexibility in choosing N, T: any FTOCP feasible for some (N, T) is also feasible for $(2N, T/2)$.

3.1.4 Experimental Results

We demonstrate the claims above using simulations and experiments on a ground rover and quadruped. The dynamical model and endogenous feedback of the unicycle model used, the high-level planner, and the low-level controller used in the experiment are described in Examples 1, 2, and 3 respectively.

Simulations. The objective is to drive a unicycle from the start to the goal location (Figure 3.2a,b). The system is subject to a matched disturbance, of magnitude $\sim 10\%$ of the control input magnitude. The reference trajectory always remains within the tightened safe set. While the unicycle's path can enter the margin between the safe set and the tightened set, it always remains safe. The value of the Lyapunov function (3.18), remains below V_{max} at all times, as expected (Figure 3.2c).

Experiments. We show the real-time efficacy and robustness of our framework by implementing it on an AION R1 UGV rover, and a Unitree A1 quadruped (Figure 3.3). For both,

[5]The only parameters the user needs to specify are N, T, l, Q, R, and γ. A line search over γ can be used to minimise the size of $\mathcal{D}$.

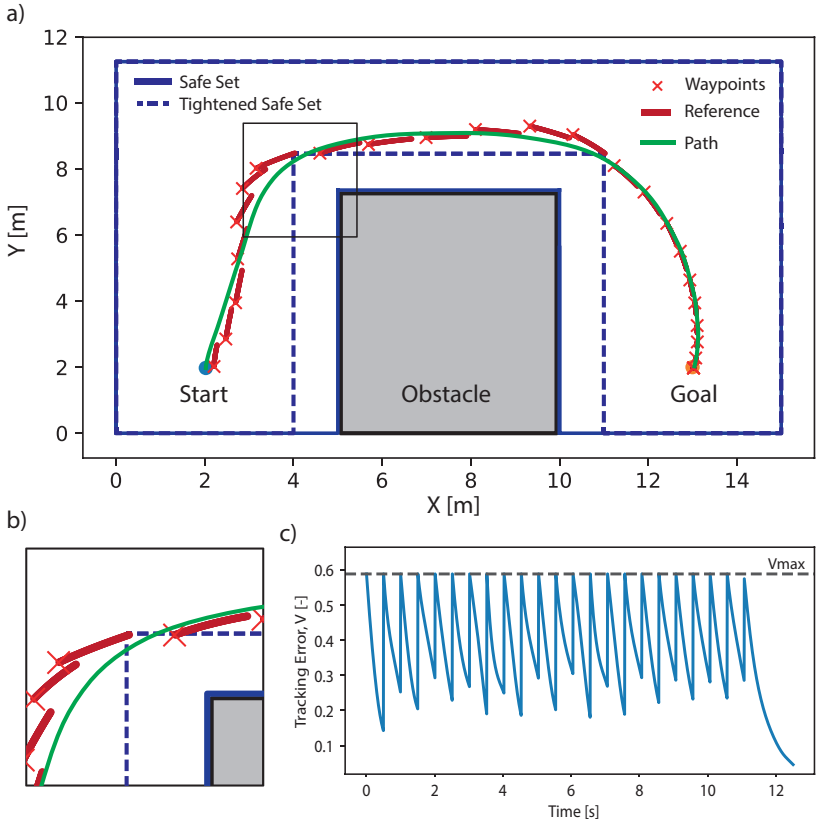

Figure 3.2: Simulation results. (a) shows unicycle (green) and reference (red) trajectories. The reference is discontinuous, since it is recomputed every T seconds. The start of each replanned reference trajectory is marked (red crosses). Black square is magnified in (b). (c) shows Lyapunov function against time, indicating that it remains below V_{max}.

the MI-QCQP MPC was implemented with `cvxpy` and solved using Gurobi. For the rover, we used $N = 9$, $T = 1.0$ seconds, and for the quadruped $N = 30$, $T = 2.0$ seconds. The low-level controllers were implemented digitally, running at 300 and 20 Hz for the rover and the quadruped respectively. Any error introduced by this sampling scheme is modeled as a part of d, the matched-disturbance to the dynamics. Each iteration took between 0.05-0.2 seconds to replan. The communication and synchronization is done with ROS. The voltage applied to the actuators are computed from the commanded velocities, by a PID for the rover, and an Inverse Dynamics Quadratic Program (ID-QP) designed using concepts in [41] for the quadruped. Our method enables safe navigation despite the presence of modelling error arising due to inability of the robots' actuators to exactly track the commanded velocities.

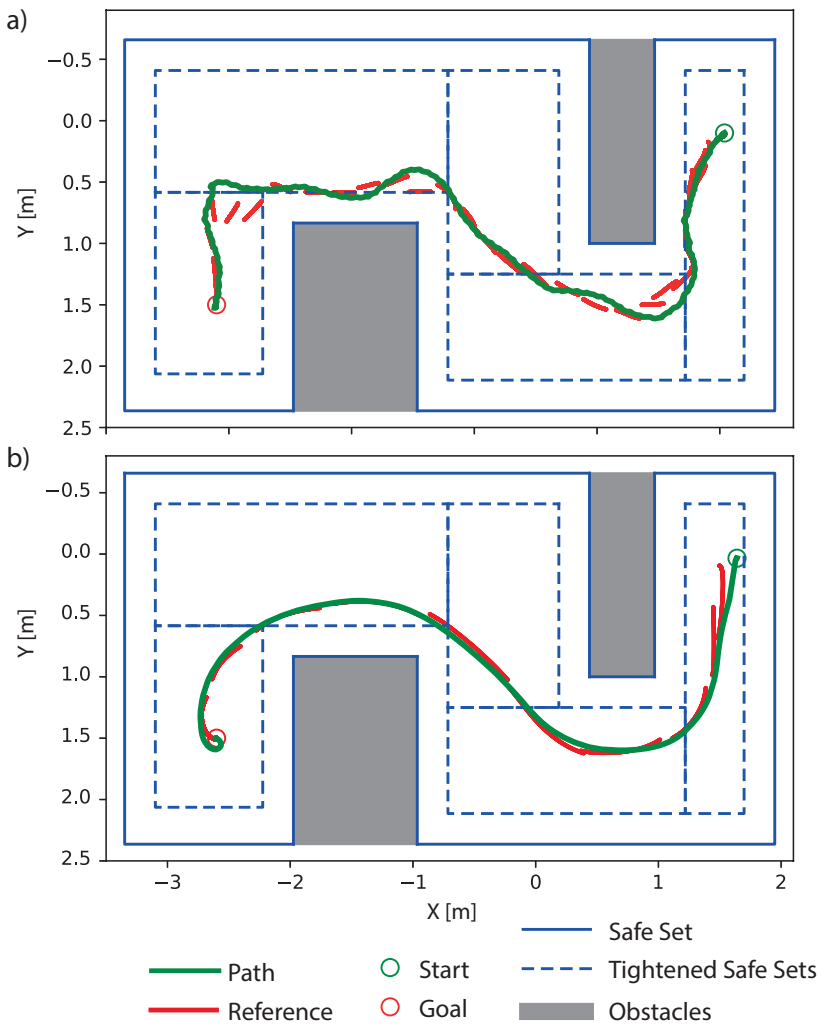

Figure 3.3: Experimental Results. The quadruped (a) and the rover (b) navigate around gray obstacles in the environment to reach target location. See Figure 3.1 for snapshots of the robots performing the experiments.

3.1.5 Conclusions

This section details a constructive method to design a multirate controller for safety-critical differentially-flat systems. The coupling between the MPC and continuous controllers allows us to claim recursive feasibility of the MPC and safety of the nonlinear system. Our theoretical claims were demonstrated in simulations and experiments. The effect of input constraints will be addressed in future works. We anticipate that penalising the flat inputs in the MPC cost function can improve input constraint satisfaction.

3.2 `gatekeeper`: A flexible framework for safe planning in online and dynamic environments

In this section, we consider the case where the safe set is not known a priori, but is rather built on-the-fly via the system outputs (sensor measurements). More specifically, we consider the problem where a robot with limited sensing capabilities (hence limited information about the environment) has to move while remaining safe under some mild assumptions on the evolution of the environment, to be stated in detail below.

Navigating within a non-convex safe set is often tackled by path planning techniques [88, 98, 138, 169]. Typically a planner generates reference (or nominal) trajectories based on a simplified (e.g., linearized or kinematic) model of the system. However, the reference trajectories may not be trackable by the actual nonlinear system dynamics, and as a result safety constraints may be violated. Furthermore, when trajectories are planned over finite horizons, without recursive feasibility guarantees a planner may fail to find a trajectories, leading to safety violations. This is particularly relevant and challenging when operating in dynamic environments.

In this section, we propose a technique to bridge path planners (that can solve the nonconvex trajectory generation problem) and controllers (that have robust stability guarantees) in a way that ensures safety. `gatekeeper` takes inspiration from [164] and [153], both of which also employ the idea of a backup planner/controller. Conceptually, a backup controller is a feedback controller that drives the system to a set of states that are safe (referred to as the backup safe set), and keeps the system in this set. For example, for a quadrotor navigating in an environment with static obstacles, a backup controller could be one that causes the quadrotor to hover in place.

In `gatekeeper`, the idea is that given a nominal trajectory generated by the path planner (potentially unsafe and/or not dynamically feasible) we construct a "committed trajectory" using a backup controller. To do this, at each iteration of `gatekeeper`, we simulate a controller that tracks the nominal trajectory upto some switching time, and executes the backup controller thereafter. The trajectory with the largest switching time that is valid (as defined in Def 3.13) becomes the committed trajectory. Thus, each committed trajectory is, by construction, guaranteed to be defined, feasible, and safe for all future time. The controller always tracks the last committed trajectory, thereby ensuring safety. This section's key contribution is the algorithm to construct such committed trajectories, and a proof that the proposed approach ensures the closed-loop system remains safe. Furthermore, we explicitly account for robustness against disturbances and state-estimation error since naive approaches to robustification can lead to undesired deadlock. The overall algorithm is computationally

efficient compared to similar methods, e.g. MPC. In our simulations 3.2.5, `gatekeeper` was approximately 3-10 times faster than MPC. `gatekeeper`'s primary limitation is that there must exist a backup controller and set. Some robotic systems and environments may not admit these components. Our focus in this section is on systems where one can find a suitable backup controller and set, and demonstrate how this can be employed to ensure safety.

In summary, this work has the following contributions:

- A framework to bridge path planners with tracking controllers in order to convert nominal/desired trajectories (generated by the path planner) into committed trajectories that the tracking controller can track safely.
- A formal proof that the robotic system will remain safe for all future time under the stated assumptions.

In particular, the new contributions with respect to [9] are:

- Theoretical: A robustification of the verification conditions in [9] to also account for state estimation errors. We have also simplified the verification conditions.
- Experimental: A demonstration of the algorithm applied to quadrotors flying through an unknown environment, constructing a map of the environment online, and filtering human pilot commands to ensure collision avoidance.

A worked analytic example is provided in the appendix, to help illustrate the key concepts of the section.

Organization

In section 3.2.1 we review a few of the leading paradigms for safety-critical path planning and control. In section 3.2.2, we describe the key idea underpinning `gatekeeper`. In sections 3.2.3, 3.2.4 we formally define the problem and describe our proposed solution. Finally, in section 3.2.5 simulations and experiments are used to demonstrate the method, and specific implementation details are discussed.

3.2.1 Related Work

A wide range of architectures and approaches have been proposed to tackle safety-critical planning and control, especially when the environment is sensed online. A generic perception planning and control stack is depicted in Fig. 3.4a.

One approach is to encode the safety constraints in the path-planning module. In this case, the world is represented using a grid-world, or through simplified geometric primitives like obstacle points, or planes to depict the walls. From this representation, a path is generated to avoid obstacles using, for instance, grid-search techniques [76] or sampling [88]. These paths can then be modified to avoid the obstacles, e.g. [107]. However since the path was generated without considering the closed-loop behavior of the nonlinear dynamics of the system and the controller, the robot may not execute the planned path exactly. Therefore, safety may not be guaranteed.

A second approach is to encode the safety constraints at the controller. In recent years, methods based on CBFs [20] have been developed to ensure that a system remains within a specified safe set while tracking a desired control input. These methods however require the safe set to be known apriori, represented by a scalar function $h : \mathcal{X} \to \mathbb{R}$ that is continuously differentiable, and satisfies an invariance condition (see for e.g. Def. 2 of [20]). For certain classes of systems and safe sets, constructive methods exist to design h, but these do not handle time-varying or multiple safety conditions well [1, 8, 37, 53, 109]. For specific system models, it is sometimes possible to construct suitable planners and controllers, e.g. [24, 46]. Alternatively, offline and computationally expensive methods based on Hamilton-Jacobi reachability (e.g. [20, 24, 25, 48, 72, 162]) or learning-based (e.g. [55, 56, 99, 108, 154]) can be used. However, when the environment is sensed online (and therefore the safe set is constructed online), the assumptions of a CBF might be difficult to verify. If unverified, these controllers could fail to maintain safety.

The third common approach is to encode safety constraints jointly between the controller and the path planner. For example, MPC plans trajectories considering the dynamics of the robotic system, and also determines a control input to track the trajectory. Various versions of this basic concept exist, e.g. [136, 143, 164, 186]. However, given the nonlinearity of the robot dynamics and the nonconvexity of the environment, guaranteeing convergence, stability or recursive feasibility is challenging. To handle the interaction between path planners and controllers, multirate controllers [8, 143] have also been proposed. These methods exploit the differential flatness of the system to provide theoretical guarantees, although the resulting mixed-integer problem can be expensive for clutter/complicated environments. In general, these methods solve the path planning problem and the control problem separately, but impose additional constraints on each to guarantee that the robot will remain safe. This assumes a structure in the path planner and the controller, limiting the applicability.

There is also a growing literature on end-to-end learning based methods for safe perception, planning, and control. See for e.g., [49, 89] and references within. These methods can perform well in scenarios that they have been trained on, but do not provide guarantees of

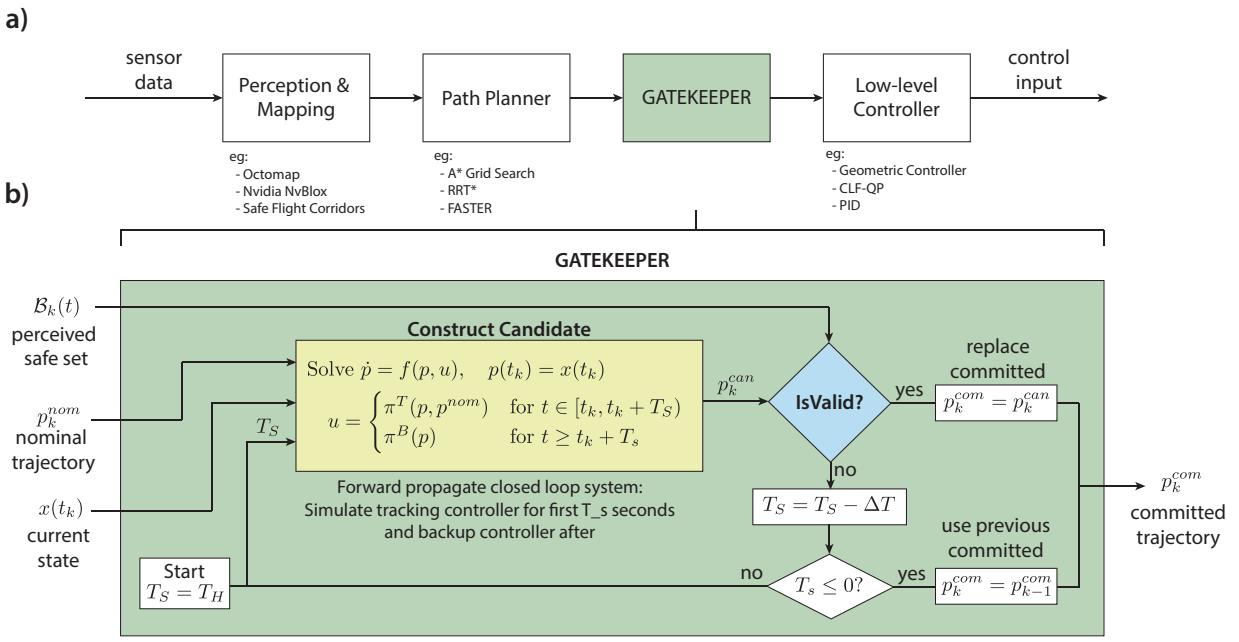

Figure 3.4: Block Diagram describing the `gatekeeper` algorithm. (a) shows that `gatekeeper` is an additional module that fits within the common perception-planning-control stack of a robotic system. (b) is a pictorial representation of Algorithm 3.1.

performance or safety in scenarios beyond which they have been trained.

The idea of backup planners/controllers has been introduced recently to address some of the above challenges. In [164], a backup trajectory is constructed using a linear model to ensure the trajectory lies within the known safe set at all times. However, since the backup trajectory was generated using simplified dynamics, the nonlinear system may not be able to execute this trajectory, possibly causing safety violations. A similar approach is proposed in [95] for mobile robots with the ability to stop. In [153], safety is guaranteed by blending the nominal and backup control inputs. The mixing fraction is determined by numerically forward propagating the backup controller. However, due to the mixing, the nominal trajectory is never followed exactly, even when it is safe to do so. By combining elements from these methods in a novel manner, `gatekeeper` addresses the respective limitations, without requiring the path planner and controller to be co-designed.

3.2.2 Motivating Example and Method Overview

We present an example to illustrate the key concepts in this section, and challenges when dealing with dynamic environments and limited sensing. A common wildfire fire-fighting mission is the “firewatch" mission, where a helicopter is deployed to trace the fire-front, the outer perimeter of the wildfire. The recorded GPS trace is then used to create a map

of the wildfire, which is then used to efficiently deploy appropriate resources. Today, the helicopters used in the firewatch mission are human-piloted, but in this example, we design an autonomous controller for a UAV to trace the fire-front without entering or being surrounded by the fire. Fig. 3.5 depicts the notation used in this section.

The fire is constantly evolving, and expanding outwards. Thus the safe set, the set of states located outside the fire, is a time-varying set denoted $\mathcal{S}(t)$. Since the rate of spread of fire is different at each location, (it depends on various environmental factors like slope, vegetation and wind [22, 144]), the evolution of the safe set $\mathcal{S}(t)$ is unknown. That said, it is often possible to bound the evolution of $\mathcal{S}(t)$. In this example, we assume the maximum fire spread rate is known. To operate in this dynamic environment, the UAV makes measurements, for example thermal images that detect the fire-front. However, due to a limited field-of-view, only a part of the safe set can be measured.

The challenge, therefore, is to design a controller for the nonlinear system that uses the on-the-fly measurements to meet mission objectives, while ensuring the system state $x(t)$ remains within the safe set at all times, i.e.,

$$x(t) \in \mathcal{S}(t),\ \forall t \geq t_0. \tag{3.30}$$

Since $\mathcal{S}$ is unknown, verifying (3.30) directly is not possible. We ask a related question: given the information available upto some time t_k, does a candidate trajectory $p_k^{can}(t)$ satisfy

$$p_k^{can}(t) \in \mathcal{B}_k(t),\ \forall t \geq t_k, \tag{3.31}$$

where $\mathcal{B}_k(t)$ is the *perceived* safe set for any time $t \geq t_k$ constructed using the sensory information available up to t_k only. If we assume the perception system provides a reliable estimate of a subset of the safe set, $\mathcal{B}_k(t) \subset \mathcal{S}(t)\ \forall t \geq t_k$, then any candidate trajectory satisfying (3.31) will also satisfy $p_k^{can}(t) \in \mathcal{S}(t)$. However, since the check in (3.31) needs to be performed over an infinite horizon $t \geq t_k$, it still cannot be implemented. *A key contribution of this section is to show how we can perform this check by verifying only a finite horizon.*

We propose the following: at each iteration, we construct a *candidate trajectory* and check whether the candidate satisfies (3.31). If so, the candidate trajectory becomes a *committed trajectory*. The controller always tracks the last committed trajectory, thus ensuring safety. In other words, the *candidate trajectory* is *valid* if it is safe over a finite horizon and reaches a backup set by the end of the horizon. The controller tracks the last valid trajectory (i.e., the committed trajectory), until a new valid trajectory is found.

Referring back to the firewatch mission, if the UAV is able to fly faster than the maximum

spread rate of the fire, a safe course of action could be to simply fly perpendicular to the firefront, i.e., radially from the fire faster than the maximum fire spread rate. This is an example of a *backup controller*, since it encodes the idea that if the system state reaches a backup set $\mathcal{C}_k(t_{kB})$ at some time $t_{kB} \geq t_k$, then the backup controller π_B^k will ensure that $x(t) \in \mathcal{C}_k(t)$ for all $t \geq t_{kB}$. Note, the notation $\mathcal{C}_k(t)$ highlights that the backup set could be a time-varying set. This switching time $t = t_k + T_s$ will be maximized by `gatekeeper` since it is off-nominal behavior.

In the firewatch mission, π_B^k is controller to make the UAV fly perpendicular to the firefront, and $\mathcal{C}_k(t)$ is the set of states that are "sufficiently far from fire, with a sufficiently high speed perpendicular to the fire." A worked example with exact expressions for $\mathcal{S}(t), \mathcal{B}_k(t), \mathcal{C}_k(t)$ is provided in the appendix. Since the fire is constantly expanding, the $\mathcal{C}_k(t)$ set is also time-varying: the set of safe states needs to be moving radially outwards. Furthermore, at each k, the backup controller and set can be a different, so we index these by k too.

Using backup controllers, we can find a sufficient condition for (3.31) that only requires finite horizon trajectories:

$$\begin{cases} p_k^{can}(t) \in \mathcal{B}_k(t) & \text{if } t \in [t_k, t_{kB}) \\ p_k^{can}(t_{kB}) \in \mathcal{C}_k(t_{kB}) \end{cases} \tag{3.32}$$

$$\implies \begin{cases} p_k^{can}(t) \in \mathcal{S}(t) & \text{if } t \in [t_k, t_{kB}) \\ p_k^{can}(t) \in \mathcal{S}(t) & \text{if } t \in [t_{kB}, \infty) \end{cases} \tag{3.33}$$

$$\iff p_k^{can}(t) \in \mathcal{S}(t)\ \forall t \geq t_k \tag{3.34}$$

for any $t_{kB} \geq t_k$, provided (I) $\mathcal{B}_k(t) \subset \mathcal{S}(t)$, (II) $\mathcal{C}_k(t) \subset \mathcal{S}(t)\ \forall t \geq t_{kB}$, and (III) for $t \geq t_{kB}$ the control input to the candidate trajectory is π_B^k. These conditions can be verified easily: (I) *is the assumption that the perception system correctly identifies a subset of the safe set*, (II) is the defining property of a backup set, and (III) will be true based on how we construct the candidate trajectory.

Notice that in (3.32), we only need to verify the candidate trajectory over a finite interval $[t_k, t_{kB}]$, but this is sufficient to proving that the candidate is safe for all $t \geq t_k$.

In the following sections, we formalize the `gatekeeper` as a method to construct safe trajectories that balance between satisfying mission objectives and ensuring safety.

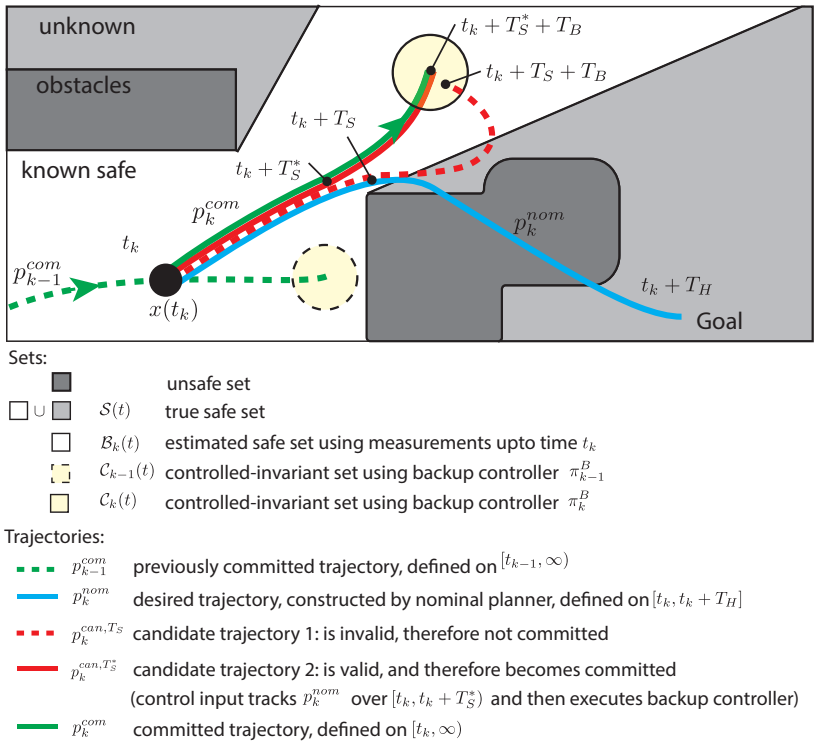

Figure 3.5: Notation used in this section. The nominal planner can plan trajectories into unknown spaces, but `gatekeeper` ensures the committed trajectory lies within the estimated safe sets, for all future time.

3.2.3 Problem Formulation

We consider two types of systems: (A) a nominal system, with perfect state information and without disturbances, and (B) a perturbed estimate-feedback system, where there are bounded disturbances on both the system dynamics and the measurements, and an observer estimates the state.

3.2.3.1 Nominal System Description

Consider a nonlinear system,

$$\dot{x} = f(x, u) \tag{3.35}$$

where $x \in \mathcal{X} \subset \mathbb{R}^n$ is the state and $u \in \mathcal{U} \subset \mathbb{R}^m$ is the control input. $f : \mathcal{X} \times \mathcal{U} \to \mathbb{R}^n$ is assumed locally Lipschitz.

Given a control policy $\pi : [t_0, \infty) \times \mathcal{X} \to \mathcal{U}$ and an initial condition $x(t_0) = x_0 \in \mathcal{X}$, the

Table 3.1: Notation

Symbol	Definition
Time Points:	
t_k	Start time of iteration k
t_{kS}	Switch time $t_{kS} = t_k + T_S$
t_{kB}	Forecast time $t_{kB} = t_{kS} + T_B$
Sets:	
$\mathcal{X} \subset \mathbb{R}^n$	State space
$\mathcal{U} \subset \mathbb{R}^m$	Control input space
$\mathcal{S}(t) \subset \mathcal{X}$	Safe set at time t
$\mathcal{B}_k(t) \subset \mathcal{X}$	Perceived safe set at time t based on measurements upto time $t_k \leq t$
$\mathcal{C}_k(t) \subset \mathcal{X}$	k-th controlled-invariant set
Controllers:	
π_T	Trajectory tracking controller, $\pi_T : \mathcal{X} \times \mathcal{X} \to \mathcal{U}$
π_B	Backup controller, $\pi_B : \mathbb{R} \times \mathcal{X} \to \mathcal{U}$
Trajectories:	
p_k^{nom}	k-th nominal trajectory
p_k^{can}	k-th candidate trajectory
p_k^{com}	k-th committed trajectory

initial-value problem describing the (nominal) closed-loop system is:

$$\dot{x} = f(x, \pi(t, x)), \qquad x(t_0) = x_0. \tag{3.36}$$

When π is piecewise continuous in t and Lipschitz wrt x, there exists an interval over which the solutions of (3.36) exist and are unique [92, Thm 3.1]. We assume this interval is $[t_0, \infty)$.

3.2.3.2 Perturbed System Description

Now consider a perturbed system without perfect state information. The perturbed system dynamics are

$$\dot{x} = f(x, u) + d(t), \tag{3.37a}$$

$$y = c(x) + v(t), \tag{3.37b}$$

where $y \in \mathbb{R}^p$ is the sensory output, and $c : \mathcal{X} \to \mathbb{R}^p$ is locally Lipschitz continuous. The additive disturbances $d : [t_0, \infty) \to \mathbb{R}^n$ and $v : [t_0, \infty) \to \mathbb{R}^p$ are bounded, $\sup_{t \geq t_0} \|d(t)\| = \bar{d} < \infty$, $\sup_{t \geq t_0} \|v(t)\| = \bar{v} < \infty$.

An observer-controller uses a state estimate $\hat{x} \in \mathcal{X}$ to compute the control input, and takes the form

$$\dot{\hat{x}} = q(\hat{x}, y, u) \tag{3.38a}$$

$$u = \pi(t, \hat{x}) \tag{3.38b}$$

where $q : \mathcal{X} \times \mathbb{R}^p \times \mathcal{U} \to \mathbb{R}^n$ is locally Lipschitz in all arguments. The estimate-feedback controller $\pi : \mathbb{R}_{\geq 0} \times \mathcal{X} \to \mathcal{U}$ is assumed piecewise-continuous in t and Lipschitz in $\hat{x}$.

In this case, the closed-loop system dynamics are:

$$\dot{\hat{x}} = q(\hat{x}, y, \pi(t, \hat{x})), \qquad \hat{x}(t_0) = \hat{x}_0, \tag{3.39a}$$

$$\dot{x} = f(x, \pi(t, \hat{x})) + d(t), \qquad x(t_0) = x_0 \tag{3.39b}$$

$$y = c(x) + v(t) \tag{3.39c}$$

We assume that for each initial $(x_0, \hat{x}_0)$ and disturbance signals d, v, a unique solution exists for all $t \in [t_0, \infty)$.

3.2.3.3 Set Invariance

Our method is based on concepts in set invariance.

Definition 3.5 (Controlled-Invariant Set)**.** *For the nominal system* (3.35)*, a controller* $\pi : [t_0, \infty) \times \mathcal{X} \to \mathcal{U}$ ***renders a set*** $\mathcal{C}(t) \subset \mathcal{X}$ ***controlled-invariant*** *on* t_0 *if, for the closed-loop system* (3.36) *and any* $\tau \geq t_0$*,*

$$x(\tau) \in \mathcal{C}(\tau) \implies x(t) \in \mathcal{C}(t), \ \forall t \geq \tau. \tag{3.40}$$

The concept of controlled invariance can be extended to the case with disturbances and an observer-controller [5].

Definition 3.6 (Robustly Controlled-Invariant Set)**.** *For the perturbed system* (3.37)*, an observer-controller* (3.38) ***renders a set*** $\mathcal{C}(t) \subset \mathcal{X}$ ***robustly controlled-invariant*** *on* t_0 *if, for the closed-loop system* (3.39) *and any bounded disturbance* d, v *with* $\sup_{t \geq t_0} \|d(t)\| \leq \bar{d}$*,* $\sup_{t \geq t_0} \|v(t)\| \leq \bar{v}$*, for any* $\tau \geq t_0$*,*

$$x(\tau) \in \mathcal{C}(\tau), \|\hat{x}(\tau) - x(\tau)\| \leq \delta \implies x(t) \in \mathcal{C}(t), \ \forall t \geq \tau. \tag{3.41}$$

for some $\delta > 0$*.*

Usually, the objective is to the find the largest controlled-invariant set $\mathcal{C}(t)$ for a given safe set $\mathcal{S}(t)$, referred to as the viability kernel [31, 48, 74]. However, these methods are difficult to apply when the safe set $\mathcal{S}(t)$ is unknown apriori, and instead is estimated online. The objective and approach of this section is different, as described below.

3.2.3.4 Assumptions

Here, we formally state the assumptions that will be used to prove that `gatekeeper` renders a system safe. We assume the following modules are available, and explain the technical assumptions of each in the following paragraphs.

1. a perception system that can sense the environment, and can estimate the safe set,
2. a nominal planner that generates desired trajectories to satisfy mission requirements (for example reaching a goal state, or exploring a region), potentially using simplified dynamic models,
3. an input-to-state stable tracking observer-controller that can robustly track a specified trajectory,
4. a backup control policy that can stabilize the system to a control invariant set.

More specifically:

Perception System

The (potentially time-varying) safe set is denoted $\mathcal{S}(t) \subset \mathcal{X}$. We assume $\mathcal{S}(t)$ always has a non-empty interior. Although the full safe set may not be known at any given time, using sensors and a model of the environment, there are scenarios in which it is possible to construct reasonable bounds on the evolution of the safe set. For example, in the firefighting scenario, an upper-bound on the fire's spread rate could be known. Similarly, in an environment with dynamic obstacles, we assume that a reasonable upper-bound on the velocity or acceleration of the dynamic obstacles is known. As such, although we address safety in unknown environment, we still require some assumptions on the behavior of the environment to guarantee safety.

Specifically, we assume that the perception system provides *estimates* of the safe set that are updated as new information is acquired by the sensors. The information is available at discrete times t_k, $k \in \mathbb{N}$. Let $\mathcal{B}_k(t)$ denote the perceived safe set for time $t \geq t_k$ constructed using sensory information upto time t_k. We assume the following:

Assumption 3.4. *The safe set $\mathcal{S}(t) \subset \mathcal{X}$ has a non-empty interior for each t, and the estimated safe set $\mathcal{B}_k(t)$ satisfies*

$$\mathcal{B}_k(t) \subset \mathcal{S}(t) \qquad \forall k \in \mathbb{N}, t \geq t_k, \tag{3.42a}$$

$$\mathcal{B}_k(t) \subset \mathcal{B}_{k+1}(t) \qquad \forall k \in \mathbb{N}, t \geq t_{k+1}. \tag{3.42b}$$

This reads as follows. In (3.42a), we assume that any state perceived to be safe is indeed safe. In (3.42b), we assume that the predictions are conservative, i.e., new information acquired at t_{k+1} *does not* reclassify a state $x \in \mathcal{B}_k(t)$ (i.e. a state perceived to be safe based on information time t_k) as an unsafe state $x \notin \mathcal{B}_{k+1}(t)$ based on information received at t_{k+1}.

This assumption (while stated more generally) is common in the literature on path planning in dynamic/unknown environments [163, 186]. Depending on the application, various methods can be used to computationally represent such sets, including SDFs [128] or SFCs [107]. If there are perception or predictions uncertainties, we assume they have already been accounted for when constructing $\mathcal{B}_k(t)$. Some methods to handle such errors are studied in [3] and references therein.

Note, Assumption 3.4 *does not* require that if a state x is classified as safe at some time t_k, that x is safe for all time. Mathematically, we *do not* assume $x \in \mathcal{B}_k(t) \implies x \in \mathcal{B}_k(\tau)\ \forall \tau \geq t$. In the appendix, diagrams and a worked example with the firefighting mission is provided to help clarify Assumption 3.4 and the definitions of $\mathcal{S}(t), \mathcal{B}_k(t)$.

Nominal Planner

We assume that a nominal planner enforces the mission requirements by specifying the desired state of the robot for a short horizon T_H into the future.

Definition 3.7 (Trajectory)**.** *A **trajectory** p with horizon T_H is a piecewise continuous function $p : \mathcal{T} \to \mathcal{X}$ defined on $\mathcal{T} = [t_k, t_k + T_H] \subset \mathbb{R}$. A trajectory p is **dynamically feasible** wrt (3.35) if there exists a piecewise continuous control $u : \mathcal{T} \to \mathcal{U}$ s.t.*

$$p(t) = p(t_k) + \int_{t_k}^{t} f(p(\tau), u(\tau))d\tau, \quad \forall t \in \mathcal{T}. \tag{3.43}$$

Denote the nominal trajectory available at the k-th iteration by the function $p_k^{nom} : [t_k, t_k + T_H] \to \mathcal{X}$. We do not require p_k^{nom} to be dynamically feasible wrt (3.35) or (3.37).

Note, although some path planners (e.g. A*, RRT*) construct geometric paths, we assume the output of the path planner is a trajectory, i.e., is parameterized by time. Methods for time allocation of geometric paths is a well studied problem, see for e.g. [107, 138, 164].

To summarize, we assume a nominal planner is available:

Assumption 3.5. *There exists a nominal planner that can generate finite-horizon trajectories* $p_k^{nom} : [t_k, t_k + T_H] \to \mathcal{X}$ *for each* $k \in \mathbb{N}$.

Tracking Observer-Controller

We assume an estimate-feedback controller $\pi_T : \mathcal{X} \times \mathcal{X} \to \mathcal{U}$ that computes a control input $u = \pi_T(\hat{x}, p(t))$ to track a given trajectory p; we refer to this policy as the tracking observer-controller [87, 100, 114]. We assume that the tracking controller is input-to-state stable [5]:

Definition 3.8 (Input-to-State Stable Observer-Controller)**.** *Let* $\mathcal{T} = [t_k, t_l] \subset \mathbb{R}_{\geq 0}$. *A tracking observer-controller*

$$u(t) = \pi_T(\hat{x}, p(t)) \tag{3.44a}$$

$$\dot{\hat{x}} = q(\hat{x}, y, u) \tag{3.44b}$$

is ***input-to-state stable*** *for the system* (3.35)*, if, for any bounded disturbances* $d : \mathcal{T} \to \mathbb{R}^n$, $v : \mathcal{T} \to \mathbb{R}$, *and any dynamically feasible trajectory* $p : \mathcal{T} \to \mathcal{X}$, *the following holds:*

$$\begin{aligned} &\|x(t_k) - \hat{x}(t_k)\| \leq \delta, \ \textit{and} \quad p(t_k) = \hat{x}(t_k) \implies \\ &\quad \|x(t) - \hat{x}(t)\| \leq \beta(\delta, t - t_k) + \gamma(\bar{w}), \ \textit{and} \\ &\quad \|\hat{x}(t) - p(t)\| \leq \beta(\delta, t - t_k) + \gamma(\bar{w}), \ \textit{and} \\ &\quad \|x(t) - p(t)\| \leq \beta(\delta, t - t_k) + \gamma(\bar{w}), \ \forall t \in \mathcal{T}, \end{aligned} \tag{3.45}$$

where $\beta : \mathbb{R}_{\geq 0} \times \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ *is class* $\mathcal{KL}$, $\gamma : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ *is class* $\mathcal{K}$, *and* $\bar{w} = \max(\sup_{t \in \mathcal{T}} \|d(t)\|, \sup_{t \in \mathcal{T}} \|v(t)\|)$.

Note, for simplicity we assumed the same β, γ for each of the three norms in (3.45), although it is not strictly necessary.

To summarize, we assume a tracking controller is known:

Assumption 3.6. *There exists an input-to-state stable observer-controller of the form in Def. 3.8, with known functions* β, γ.

Backup Controller

In the case when a safe set $\mathcal{S}$ can not be rendered controlled invariant for given system dynamics, the objective reduces to finding a set $\mathcal{C} \subset \mathcal{S}$, and a controller $\pi : \mathcal{C} \to \mathcal{U}$ that renders $\mathcal{C}$ controlled invariant. For example, by linearizing (3.35) around a stabilizable equilibrium x_e, an LQR controller renders a (sufficiently small) set of states around x_e

forward invariant [92, Thm. 4.13, 4.18]. This observation leads to the notion of backup safety [47, 153].

Definition 3.9 (Backup Controller)**.** *A controller $\pi_B^k : \mathcal{T} \times \mathcal{X} \to \mathcal{U}$ is a **backup controller** to a set $\mathcal{C}_k(t) \subset \mathcal{X}$ defined for $t \in \mathcal{T} = [t_k, \infty)$ if, for the closed-loop system*

$$\dot{x} = f(x, \pi_B^k(t, x)), \tag{3.46}$$

(A) there exists a neighborhood $\mathcal{N}_k(t) \subset \mathcal{X}$ of $\mathcal{C}_k(t)$, s.t. $\mathcal{C}_k(t)$ is reachable in fixed time T_B:

$$x(\tau) \in \mathcal{N}_k(\tau) \implies x(\tau + T_B) \in \mathcal{C}(\tau + T_B), \tag{3.47}$$

and (B) π_B^k renders $\mathcal{C}_k(t)$ controlled-invariant:

$$x(\tau + T_B) \in \mathcal{C}(\tau + T_B) \implies x(t) \in \mathcal{C}(t)\ \forall t \geq \tau + T_B. \tag{3.48}$$

Remark 3.5. *The neighborhood $\mathcal{N}_k$ does not need to be known in the `gatekeeper` framework. The definition ensures that there exist states outside $\mathcal{C}_k$ that can be driven into $\mathcal{C}_k$ by the backup controller within a fixed time T_B. This excludes cases, for example, where the backup trajectories approach $\mathcal{C}_k$ asymptotically but never actually enter $\mathcal{C}_k$.*

To summarize, we assume a backup controller is known:

Assumption 3.7. *At the k-th iteration, a set $\mathcal{C}_k(t)$ and a backup controller $\pi_B^k : [t_k, \infty) \times \mathcal{X} \to \mathcal{U}$ to $\mathcal{C}_k(t)$ can be found where*

$$\mathcal{C}_k(t) \subset \mathcal{S}(t),\ \forall t \geq t_k. \tag{3.49}$$

Remark 3.6. *Note that while we assume $\mathcal{C}_k(t) \subset \mathcal{S}(t)$, we* do not *assume the trajectory to reach $\mathcal{C}_k(t)$ is safe, nor that the set $\mathcal{C}_k(t)$ is reachable from the current state $x(t_k)$ within a finite horizon. This is in contrast to backward reachability based methods [1, 73, 109, 162]. Instead, we will ensure both of these conditions are satisfied through our algorithm.*

Remark 3.7. *The design of backup controllers and sets depends on the robotic system and the environment model. For some systems, the backup set can be designed by linearization about a stabilizable equilibrium point (or limit cycle), and determining the region of attraction. Other methods include reachability analysis or learning-based approaches, e.g. [25, 55, 56, 99, 108, 154]. Generic methods to design the backup controllers are beyond the scope of this section, but specific methods are discussed in Section 3.2.5 and in [47, 153].*

3.2.3.5 Problem Statement

In summary, the problem statement is

Problem 3.1. *Consider system* (3.37) *satisfying Assumptions 3.4-3.7, i.e., a system with a perception system satisfying Assumption 3.4, a nominal planner that generates desired trajectories, an input-to-state stable tracking controller satisfying Definition 3.8, and a backup controller satisfying Assumption 3.7. Design an algorithm to track desired trajectories while ensuring safety, i.e.,* $x(t) \in \mathcal{S}(t)$ *for all* $t \geq t_0$.

3.2.4 Proposed Solution

`gatekeeper` is a module that lies *between* the planning and control modules. It considers the nominal trajectories by the planner, modifies them as needed to what we call committed trajectories, and inputs these committed trajectories to the trajectory-tracking controller. In this section, we will demonstrate how to construct these committed trajectories. To aid the reader, the analysis is first presented for the nominal case, and later extended to the perturbed case. The various trajectories and times are depicted in Fig. 3.5. The algorithm is described in Algorithm 3.1 and depicted in Fig. 3.4.

3.2.4.1 Nominal Case

At the k-th iteration, $k \in \mathbb{N} \setminus \{0\}$, let the previously committed trajectory be p_{k-1}^{com}. `gatekeeper` constructs a candidate trajectory p_k^{can,T_S} by forward propagating a controller that tracks p_k^{nom} over an interval $[t_k, t_k + T_S)$, and executes the backup controller for $t \geq t_k + T_S$. $T_S \in \mathbb{R}_{\geq 0}$ is a switching duration maximized by `gatekeeper` as described later. Formally,

Definition 3.10 (Candidate Trajectory)**.** *Suppose at* $t = t_k$,

- *the state is* $x(t_k) = x_k$,
- *the nominal trajectory is* $p_k^{nom} : [t_k, t_k + T_H] \to \mathcal{X}$,
- π_T *is a trajectory tracking controller,*
- π_B^k *is a backup controller to the set* $\mathcal{C}_k(t)$.

*Given a $T_S \in [0, T_H]$, the **candidate trajectory** $p_k^{can,T_S} : [t_k, \infty) \to \mathcal{X}$ is the solution to the initial value problem*

$$\dot{p} = f(p, u(t)), \tag{3.50a}$$

$$p(t_k) = x_k, \tag{3.50b}$$

$$u(t) = \begin{cases} \pi_T(p(t), p_k^{nom}(t)) & t \in [t_k, t_k + T_S) \\ \pi_B^k(t, p(t)) & t \geq t_k + T_S. \end{cases} \tag{3.50c}$$

By construction, the candidate is dynamically feasible wrt (3.35). A candidate trajectory is *valid* if the following hold:

Definition 3.11 (Valid)**.** *A candidate trajectory $p_k^{can,T_s} : [t_k, \infty) \to \mathcal{X}$ defined by (3.50) is **valid** if the trajectory is safe wrt the estimated safe set over a finite interval:*

$$p_k^{can,T_S}(t) \in \mathcal{B}_k(t), \ \forall t \in [t_k, t_{k,B}], \tag{3.51}$$

and the trajectory reaches $\mathcal{C}_k(t)$ at the end of the horizon:

$$p_k^{can,T_S}(t_{k,B}) \in \mathcal{C}_k(t_{k,B}), \tag{3.52}$$

where $t_{k,B} = t_k + T_S + T_B$.

Notice checking whether a candidate is valid only requires the solution p_k^{can,T_S} over the finite interval $[t_k, t_k + T_S + T_B]$. This means that the candidate can be constructed by numerical forward integration over a finite horizon.

Def. 3.12 defines how the k-th committed trajectory is constructed using the nominal trajectory p_k^{nom}, the backup controller π_k^B, and the previous committed trajectory p_{k-1}^{com}.

Definition 3.12 (Committed Trajectory)**.** *At the k-th iteration, define*

$$\mathcal{I}_k = \left\{ T_S \in [0, T_H] : p_k^{can,T_S} \text{ is valid} \right\} \subset \mathbb{R}, \tag{3.53}$$

where $p_k^{can,T_S} : [t_k, \infty) \to \mathcal{X}$ is as defined in (3.50), and Def. 3.11 is used to check validity. The committed trajectory is $p_k^{com} : [t_k, \infty) \to \mathcal{X}$, defined as follows:

*If $\mathcal{I}_k \neq \varnothing$, let $T_S^* = \max \mathcal{I}_k$. The **committed trajectory** is*

$$p_k^{com}(t) = p_k^{can,T_S^*}(t), \quad t \in [t_k, \infty). \tag{3.54}$$

If $\mathcal{I}_k = \varnothing$, the ***committed trajectory*** *is*

$$p_k^{com}(t) = p_{k-1}^{com}(t), \quad t \in [t_k, \infty). \tag{3.55}$$

We are ready to prove the proposed strategy guarantees safety. First, we show that each committed trajectory is safe.

Theorem 3.6. *Suppose Assumptions 3.4-3.7 hold. Suppose $p_0^{can,T_S} : [t_0, \infty) \to \mathcal{X}$ is a candidate trajectory that is dynamically feasible wrt* (3.35) *and valid according to Def. 3.11 for some $T_S \geq 0$. If, for every $k \in \mathbb{N}$, $p_k^{com} : [t_k, \infty) \to \mathcal{X}$ is determined using Def. 3.12, then for all $k \in \mathbb{N}$,*

$$p_k^{com}(t) \in \mathcal{S}(t), \quad \forall t \in [t_k, \infty). \tag{3.56}$$

Proof. The proof is by induction.

Base Case: $k = 0$. Since p_0^{can} is a valid trajectory, it is committed, i.e., $p_0^{com} = p_0^{can,T_S}$. Then,

$$\begin{aligned} p_0^{com}(t) &\in \begin{cases} \mathcal{B}_0(t) & \text{for } t \in [t_0, t_{0,B}) \\ \mathcal{C}_0(t) & \text{for } t = t_{0,B} \end{cases} \\ \implies p_0^{com}(t) &\in \begin{cases} \mathcal{S}(t) & \text{for } t \in [t_0, t_{0,B}) \\ \mathcal{S}(t) & \text{for } t \geq t_{0,B} \end{cases} \\ \iff p_0^{com}(t) &\in \mathcal{S}(t) \text{ for } t \geq t_0 \end{aligned}$$

where $t_{0,B} = t_0 + T_S + T_B$.

Induction Step: Suppose the claim is true for some $k \in \mathbb{N}$. We will show the claim is also true for $k+1$. There are two possible definitions for p_k^{com}:

Case 1: When $\mathcal{I}_{k+1} \neq \varnothing$, p_{k+1}^{can,T_S^*} is a valid candidate, i.e.,

$$\begin{aligned} p_{k+1}^{com}(t) &= p_{k+1}^{can,T_S^*}(t) \ \forall t \geq t_0 \\ &\in \begin{cases} \mathcal{B}_{k+1}(t) & \text{for } t \in [t_{k+1}, t_{k+1,SB}) \\ \mathcal{C}_{k+1}(t) & \text{for } t \geq t_{k+1,SB} \end{cases} \\ &\in \mathcal{S}(t) \text{ for } t \geq t_{k+1} \end{aligned}$$

Case 2: If $\mathcal{I}_{k+1} = \varnothing$, the committed trajectory is unchanged,

$$p_{k+1}^{com}(t) = p_k^{com}(t) \in \mathcal{S}(t),\ \forall t \geq t_{k+1}.$$

□

The following shows that `gatekeeper` ensures safety.

Theorem 3.7. *Under the assumptions of Theorem 3.6, if* $x(t_0) = p_0^{com}(t_0)$*, and for each* $k \in \mathbb{N}$ *the control input to the nominal system* (3.35) *is*

$$u(t) = \pi_T^k(x(t), p_k^{com}(t)), \forall t \in [t_k, t_{k+1}), \tag{3.57}$$

then the closed-loop dynamics (3.36) *will satisfy*

$$x(t) \in \mathcal{S}(t), \forall t \geq t_0. \tag{3.58}$$

Proof. We prove this by showing that $\forall k \in \mathbb{N}$, $x(t) = p_k^{com}(t)$ for $t \in [t_k, t_{k+1})$. Again, we use induction.

Base Case: For the nominal system (3.35), when $x(t_0) = p_0^{com}(t_0)$ and the tracking controller is ISS (3.45),

$$\begin{aligned} \|x(t) - p_0^{com}(t)\| &\leq \beta(0, t - t_0) + \gamma(0) = 0 \\ &\therefore x(t) = p_0^{com}(t)\ \forall t \in [t_0, t_1) \end{aligned}$$

Induction Step: Suppose for some $k \in \mathbb{N}$, $x(t) = p_k^{com}(t)$ for $t \in [t_k, t_{k+1})$. There are two cases for p_{k+1}^{com}:

Case 1: a new candidate is committed, $\therefore p_{k+1}^{can,T_S}(t_{k+1}) = x(t_{k+1})$. Since the tracking controller is input-to-state stable, this implies $x(t) = p_{k+1}^{com}(t)$ for $t \in [t_{k+1}, t_{k+2})$.

Case 2: A new candidate is not committed, $\therefore p_{k+1}^{com}(t) = p_k^{com}(t)\ \forall t \in [t_{k+1}, t_{k+2})$. Since $x(t_{k+1}) = p_k^{com}(t_{k+1})$, the tracking controller ensures $x(t) = p_{k+1}^{com}(t)\ \forall t \in [t_{k+1}, t_{k+2})$.

Therefore, $x(t) = p_k^{com}(t) \in \mathcal{S}(t)\ \forall t \in [t_k, t_{k+1})$, for each $k \in \mathbb{N}$. Thus, $x(t) \in \mathcal{S}(t)$ for all $t \geq t_0$. □

Remark 3.8. *The controller in* (3.57) *uses the backup controller* π_B*: the committed trajectory* p_k^{com} *is constructed such that for all* $t \geq t_{k,SB}$ *the trajectory uses the backup controller (see* (3.50)*). Therefore, if after the* k*-th step new candidate trajectories are not committed, the controller* (3.57) *applies the backup controller for time* $t \geq t_{k,SB}$*.*

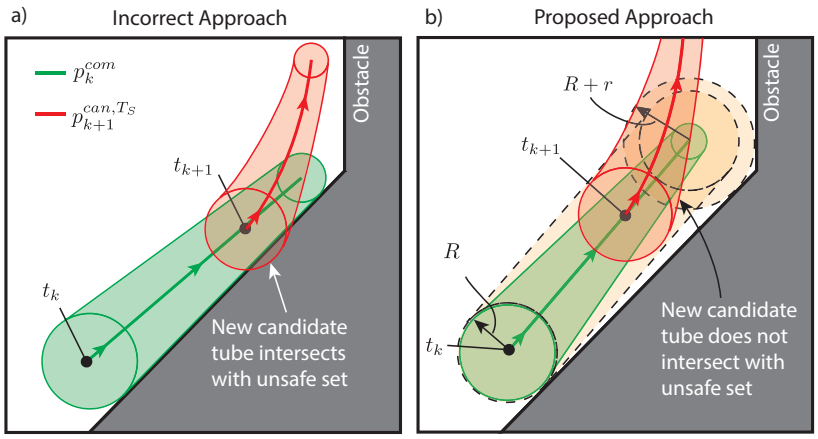

Figure 3.6: Diagram depicting the challenge due to disturbances. (a) Green line shows the committed trajectory at iteration k, and the shaded region is the tube that contains the system trajectory. If the validation step only checks that the green tube lies within the safe set, a new candidate trajectory (red) cannot be committed, since the candidate tube (red shaded region) intersects with the unsafe set. (b) shows the proposed approach, where safety is checked wrt the yellow set, i.e., a tube of radius R along the trajectory and a ball of radius $R + r$ at the end. This allows for sufficient margin to commit a new trajectory at the next iteration.

Remark 3.9. *In [47, 153], numerical forward propagation of the trajectory with a backup controller is also used to construct a safety filter. However, the resulting control input mixes the nominal control input with the backup control input at all times. In contrast, in* `gatekeeper` *we use a switching time to switch between implementing the nominal control input and the backup control input. This is desirable since it leads to less conservative controllers, as highlighted in section 3.2.5.1.*

3.2.4.2 Perturbed Case

We now address the case with non-zero disturbances and state-estimation error.[6] The algorithm is identical to that presented above, except that the validation step will be redefined.

First, we highlight the problem that disturbances introduce. Consider the specific scenario visualized in Fig. 3.6. To account for the disturbances, we validate safety of a tube around the candidate trajectory: using the ISS bound (3.45), a tube of decreasing radius around the committed trajectory will always contain the true state of the system. Therefore, if instead of (3.51), we checked that the corresponding tube containing the candidate trajectory lies within the safe set (green tube in Fig. 3.6a), then indeed, the system will remain safe.

[6]Compared to the conference version [9], here we consider the additional uncertainty due to state estimation errors, and simplify the validation check.

However, when a new candidate is proposed at the next iteration, the new tube (red tube) intersects with the unsafe set. Thus, the new candidate cannot be committed, and an undesired deadlock is reached: $x(t) \in \mathcal{C}_k(t)$ for all $t \geq t_{k,B}$.

To avoid this behavior, we propose a different validity check. First, we check that a tube of radius R is safe over the finite horizon, and second, we ensure $\mathcal{C}_k$ is a (larger) distance $R + r$ away from the unsafe set, where r, R are defined below. In Fig. 3.6a, this is depicted by the yellow sets. Note, the additional $+r$ term is used to avoid the described deadlock behaviour, and is not needed to guarantee safety.

Recall Def. 3.8 defines the controller's tracking error bounds. The validity check in Def. 3.11 is replaced by the following:

Definition 3.13 (Robustly Valid)**.** *Consider the dynamical system* (3.37)*, with bounded disturbances* $\sup_{t \geq t_k} \|d(t)\| \leq \bar{d}$*, and* $\sup_{t \geq t_k} \|v(t)\| \leq \bar{v}$*. Let* $\bar{w} = \max(\bar{d}, \bar{v})$*. Suppose* $\|x(t_k) - \hat{x}(t_k)\| \leq r$ *for some* $k \in \mathbb{N}$*. Let* $R = \beta(r, 0) + \gamma(\bar{w})$*.*

A candidate trajectory $p_k^{can,T_s} : [t_k, \infty) \to \mathcal{X}$ *defined by* (3.50) *is* ***robustly valid*** *if*

- *the candidate trajectory coincides with the state estimate at the initial time:*

$$\hat{x}(t_k) = p_k^{can,T_s}(t_k), \tag{3.59}$$

- *the candidate trajectory is robustly safe over a finite interval:*

$$p_k^{can,T_S}(t) \in \mathcal{B}_k(t) \ominus \mathbb{B}(R) \ \forall t \in [t_k, t_{k,B}], \tag{3.60}$$

- *at the end of the interval, it reaches* $\mathcal{C}_k(t)$*:*

$$p_k^{can,T_S}(t_{k,SB}) \in \mathcal{C}_k(t_{k,B}), \tag{3.61}$$

- *and the set* $\mathcal{C}_k(t)$ *is* $(R + r)$ *away from the unsafe set:*

$$\mathcal{C}_k(t) \subset \mathcal{S}(t) \ominus \mathbb{B}(R + r) \ \forall t \geq t_k. \tag{3.62}$$

If a candidate trajectory is robustly valid, it can be committed. The following theorem proves that `gatekeeper` can render the perturbed system (3.37) safe.

Theorem 3.8. *Suppose Assumptions 3.4-3.7 hold. Suppose* $p_0^{com} : [t_0, \infty) \to \mathcal{X}$ *is a committed trajectory that is* robustly *valid by Def. 3.13 for some* $r > 0, T_S \geq 0$*. Suppose* $\|x(t_0) - \hat{x}(t_0)\| \leq r$*, and* $p_0^{com}(t_0) = \hat{x}(t_0)$*.*

If, for every $k \in \mathbb{N}\backslash\{0\}$, $p_k^{com} : [t_k, \infty) \to \mathcal{X}$ is determined using Def. 3.12 (except that validity is checked using Def. 3.13), and the control input to the perturbed system (3.37) *is*

$$u(t) = \pi_T^k(\hat{x}(t), p_k^{com}(t)) \quad \forall t \in [t_k, t_{k+1}]$$

then the closed-loop system (3.39) *will satisfy*

$$x(t) \in \mathcal{S}(t), \quad \forall t \geq t_0. \tag{3.63}$$

Proof. As in Thm. 3.6, we have that for any $k \in \mathbb{N}$,

$$p_k^{com}(t) \in \mathcal{S}(t) \quad \forall t \in [t_k, \infty). \tag{3.64}$$

We aim to prove the analog of Thm 3.7, i.e., that for any $k \in \mathbb{N}$, tracking the committed trajectory $p_k^{com}(t_k)$ for $t \geq t_k$ is safe. This is proved below.

Since p_k^{com} is robustly valid, $p_k^{com}(t_k) = \hat{x}(t_k)$. Therefore,

$$\|x(t_k) - p_k^{com}(t_k)\| = \|x(t_k) - \hat{x}(t_k)\| \leq r.$$

Using (3.45), this implies that for all $t \geq t_k$,

$$\begin{aligned} \|x(t) - p_k^{com}(t)\| &\leq \beta(r, t - t_k) + \eta(\bar{w}) \\ &\leq \beta(r, 0) + \eta(\bar{w}) = R \\ \therefore \|x(t) - p_k^{com}(t)\| &\leq R \end{aligned}$$

and so by (3.60),

$$\begin{aligned} & p_k^{com}(t) \in \mathcal{B}_k(t) \ominus \mathbb{B}(R) && \forall t \in [t_k, t_{k,SB}] \\ \implies & \{p_k^{com}(t)\} \oplus \mathbb{B}(R) \subset \mathcal{B}_k(t) && \forall t \in [t_k, t_{k,SB}] \\ \implies & x(t) \in \mathcal{B}_k(t) && \forall t \in [t_k, t_{k,SB}]. \end{aligned}$$

Furthermore, since for all $t \geq t_k + T_S$ the committed trajectory is generated by the backup controller, and $p_k^{com}(t_{k,B}) \in \mathcal{C}_k(t_{k,B})$, we have $p_k^{com}(t) \in \mathcal{C}_k(t), \forall t \geq t_{k,B}$. Therefore,

$$x(t) \in \mathcal{C}_k(t) \oplus \mathbb{B}(R) \quad \forall t \geq t_{k,B}.$$

Putting these together,

$$
\begin{aligned}
x(t) &\in \begin{cases} \mathcal{B}_k(t) & \text{for } t \in [t_k, t_{k,B}] \\ \mathcal{C}_k(t) \oplus \mathbb{B}(R) & \text{for } t \geq t_{k,B} \end{cases} \\
\implies x(t) &\in \begin{cases} \mathcal{S}_k(t) & \text{for } t \in [t_k, t_{k,B}) \\ \mathcal{S}_k(t) & \text{for } t \geq t_{k,B} \end{cases} \\
\iff x(t) &\in \mathcal{S}(t) \quad \forall t \geq t_k.
\end{aligned}
$$

□

This proves that for any $k \in \mathbb{N}$, if p_k^{com} is the committed trajectory, the system will remain safe while it is tracking p_k^{com}. When a new candidate trajectory that is robustly valid (by Def. 3.13) is found, the committed trajectory can be updated, and the system will continue to remain safe.

Remark 3.10. *The theorem provides certain parameters of the nominal planner. For instance, requiring trajectories to lie in $\mathcal{B}(t_k) \ominus \mathbb{B}(R)$ corresponds to the common practice of inflating the unsafe sets by a radius R. The theorem shows that any $R \geq \beta(r, 0) + \gamma(\bar{w})$ is sufficient.*

Remark 3.11. *In (3.62), we checked that $\mathcal{C}_k(t)$ is at least $(R + r)$ away from the boundary of $\mathcal{S}(t)$ at all $t \geq t_k$, even though the proof of safety only requires a margin R. The reason we check for $(R + r)$ is to prevent the deadlock scenario discussed before: under the stated assumptions, for $t \geq t_k$, $\|x(t) - \hat{x}(t)\| \leq \beta(t - t_k) + \gamma(\bar{w})$. Therefore, if $r \geq \gamma(\bar{w})$ there exists some time $\tau = t_k + T$ where $r = \beta(T, r) + \gamma(\bar{w})$ since class $\mathcal{KL}$ functions are strictly decreasing wrt t. Thus, for $t \geq \tau$, $p_k^{com}(t) \in \mathcal{C}_k(t)$. Thus,*

$$x(t), \hat{x}(t) \in \mathcal{C}_k(\tau) \oplus \mathbb{B}(r)$$

and $\|x(t) - \hat{x}(t)\| \leq r$. Thus, when validating the new candidate trajectory $p_{k'}^{can,T_s}$ they will start at least R away from the boundary, i.e., there is sufficient margin for new trajectories to be committed.

Remark 3.12. *In constructing candidate trajectories, we require the initial state of the candidate trajectory to coincide with the state estimate, (3.59). If this is not the case, an additional margin would be necessary in (3.62) to account for this error.*

Remark 3.13. *The construction of committed trajectories is summarized in pseudo-code in Alg. 3.1.* $\max \mathcal{I}$ *can be determined efficiently, since it is an optimization of a scalar variable*

over a bounded interval. We used a simple grid search with N points. Therefore, upto N initial value problems need to be solved, which can be done very efficiently using modern solvers [134]. Using $N = 10$, the median computation time was only 3.4 ms. Other strategies including log-spacing or optimization techniques could be investigated in the future.

Algorithm 3.1: `gatekeeper`

Parameters: $N > 0 \in \mathbb{N}$
`// Do a grid search backwards over the interval` $[0, T_H]$`:`
for *i in range(0, N):* **do**
  Using $\mathcal{B}_k(t)$, identify $\mathcal{C}_k(t)$ satisfying assum. 3.7.
  $T_S = (1 - i/N)T_H$
  Solve the initial value problem (3.50) to determine $p_k^{can,T_S}(t)$ over the interval $[t_k, t_k + T_S + T_B]$
  if p_k^{can,T_S} *is robustly valid by Def. 3.13* **then**
    $p_k^{com} = p_k^{can,T_S}$
    return
`// no candidate is valid,` $\mathcal{I} = \varnothing$
$p_k^{com} = p_{k-1}^{com}$
return

3.2.5 Simulations and Experiments

Code and videos are available here: https://github.com/dev10110/gatekeeper. We test two case studies to evaluate `gatekeeper`, where the second case study is also performed using hardware experiments. A key strength of `gatekeeper` is that it can be composed with existing perception, planning and control algorithms, and the various techniques used are summarized in Table 3.2. The details are provided in the following paragraphs.

3.2.5.1 Firewatch Mission

We simulate an autonomous helicopter performing the firewatch mission, around a fire with an initial perimeter of 16 km. The helicopter starts 0.45 km from the fire, and is tasked to fly along the perimeter at a target airspeed of 15 m/s without entering the fire. The helicopter

Table 3.2: Methods used in implementing `gatekeeper` for each case study. Details are provided in the text.

	Firewatch Mission	Quadrotor Navigation
Sensed Data	Image of fire	RGBD image
Perception Output	SDF	SDF + SFC
Nominal Planner	MPC	Distance Map Planner (DMP)
Tracking Controller	PD-Controller	Geometric Controller
Backup Controller	Fly perpendicular to fire	Stop and yaw

Table 3.3: Comparison of `gatekeeper` (ours) with the nominal planner, FASTER [164], and backup filters [153]. The distance to the firefront, velocity of the helicopter, and computation time per iteration are reported for each method. IQR = interquartile range. *Since the backup filter is run at each control iteration instead of every planning iteration, it runs 20 times as often as `gatekeeper`, i.e., is 5 times as computationally expensive as `gatekeeper`.

	Distance to Fire [km]			Speed [m/s]		Comp. time [ms]		
	Minimum	Mean	Std.	Mean	Std.	Median	IQR	
Target	≥ 0	0.100	-	15.0	-	-	-	
Nominal Planner	-0.032	0.098	0.032	15.14	0.73	27.32	4.37	Unsafe
FASTER [164]	0.040	0.101	0.030	12.60	2.08	78.50	20.64	Safe, but gets trapped in pocket
Backup Filters [153]	0.081	0.240	0.054	10.11	3.52	0.87*	0.05	Safe, but conservative and slow
Gatekeeper (proposed)	0.049	0.108	0.034	14.91	1.35	3.39	0.11	Safe

is modeled as

$$\begin{aligned}\dot{x}_1 &= x_3 \cos x_4 \\ \dot{x}_2 &= x_3 \sin x_4 \\ \dot{x}_3 &= u_1 \\ \dot{x}_4 &= (g/x_3) \tan u_2,\end{aligned}$$

where x_1, x_2 are the Cartesian position coordinates of the helicopter wrt an inertial frame, x_3 is the speed of the vehicle along its heading, x_4 is the heading, and g is the acceleration due to gravity. The control inputs are u_1, the acceleration along the heading, and u_2, the roll angle. The inputs are bounded, with $|u_1| < 0.5g$ and $|u_2| < \pi/4$ rad. This system models a UAV that can control its forward airspeed and makes coordinated turns. Notice the model has a singularity at $x_3 = 0$, and the system is *not* control affine.

The fire is modeled using level-set methods [16]. In particular, the fire is described using the implicit function $\phi : \mathbb{R} \times \mathbb{R}^2 \to \mathbb{R}$, where $\phi(t, p)$ is the signed distance to the firefront

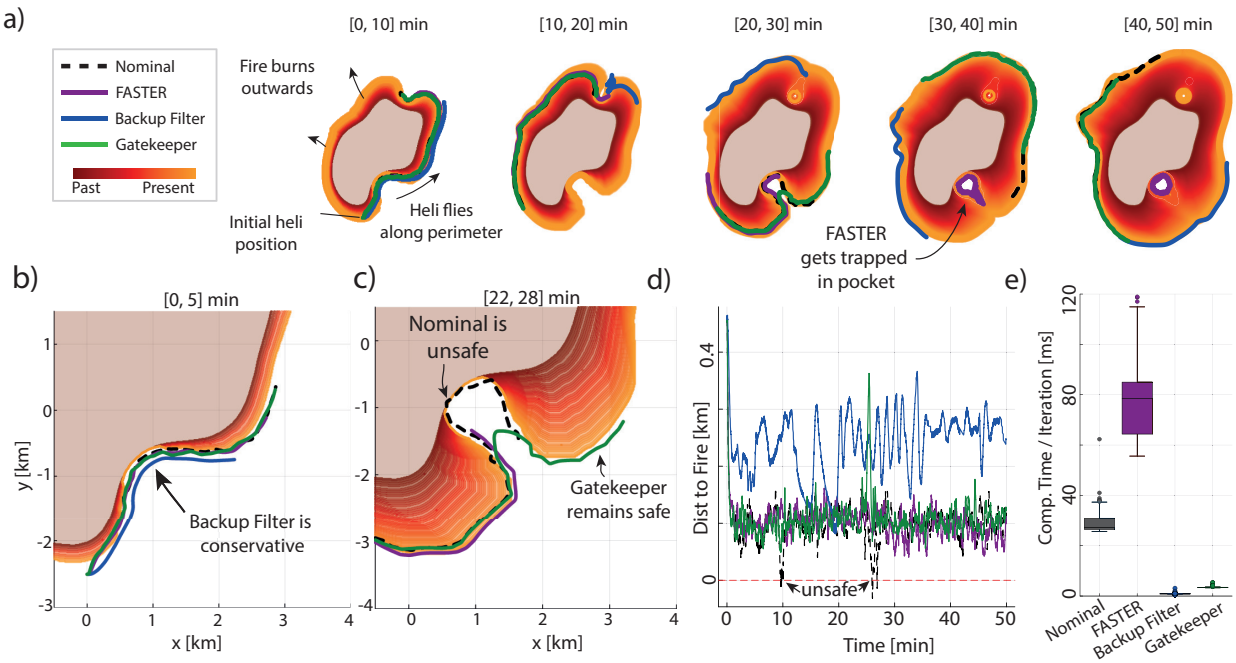

Figure 3.7: Simulation results from Firewatch mission. (a) Snapshots of the fire and trajectories executed by each of three controller. The fire is spreading outwards, and the helicopters are following the perimeter. The black line traces the nominal controller, the blue line is based on the backup filter adapted from [153] and the green line shows the proposed controller. (b, c) show specific durations in greater detail. At $t = 0$, the `gatekeeper` controller behaves identically to the nominal controller, and makes small modifications when necessary to ensure safety. The backup filter is conservative, driving the helicopter away from the fire and slowing it down. (d) Plot of minimum distance to fire-front across time for each of the controllers. (e) The nominal controller becomes unsafe 3 times, while FASTER, the backup controller, and the `gatekeeper` controllers maintain safety. Animations are available at https://github.com/dev10110/gatekeeper.

from location p at time t. Hence, the safe set is

$$\mathcal{S}(t) = \{x : \phi(t, [x_1, x_2]^T) \geq 0\}$$

where $[x_1, x_2]$ are the Cartesian coordinates of the UAV.

The evolution of the fire is based on the Rothermel 1972 model [144]. Given the Rate of Spread (RoS) function $\sigma : \mathbb{R}^2 \to \mathbb{R}$, the safe set evolves according to

$$\frac{\partial \phi}{\partial t}(t, p) + \sigma(p) \left\| \nabla \phi(t, p) \right\| = 0 \; \forall p \in \mathbb{R}^2 \tag{3.66}$$

The RoS depends on various environmental factors including terrain topology, vegetation type, and wind [22, 144] but can be bounded [54]. The simulated environment used a RoS

function that the controllers did not have access to. The only information the controllers could use was the thermal image (to detect the fire within a ± 1 km range of the UAV) and the assumption that the maximum rate of spread is 8 km/h.

We compare our approach against the nominal planner and two state of the art methods for similar problems, Fig. 3.7, Table 3.3. In particular, we compare (A) a nominal planner (black), (B) FASTER [164] (purple), (C) Backup Filters [153] (blue) and (D) `gatekeeper` (green). Since these methods were not originally developed for dynamic environments with limited sensing, both methods (B, C) were modified to be applicable to this scenario. See https://github.com/dev10110/gatekeeper for details. Method (A) represents the baseline planner without any safety filtering. Methods (B), (C) and (D) are safety filtering methods that use the nominal planner of (A) and modify it to ensure safety.

The simulation environment and each of the methods were implemented in `julia`, to allow for direct comparison, using `Tsit5()` [134] with default tolerances. Each run simulates a flight time of 50 minutes. The tracking controller was implemented as zero-order hold, updated at 20 Hz. Measurements of the firefront were available at 0.1 Hz, triggering the planners to update, intentionally slow to highlight the challenges of slow perception/planning systems. The measurements are a bitmask image, defining the domain where $\phi \leq 0$, at a grid resolution of 10 meters. These simulations were performed on a 2019 Macbook Pro (Intel i9, 2.3 GHz, 16 GB).

In the nominal planner, a linear MPC problem is solved to generate trajectories that fly along the local tangent 0.1 km away from firefront at 15 m/s. The planner uses a simplified dynamic model, a discrete-time double integrator. This convex quadratic program (QP) is solved using `gurobi`. The median computation time is 27 ms, using $N = 40$ waypoints and a planning horizon of 120 seconds. The tracking controller is a nonlinear feedback controller based on differential flatness [8, 114]. When tracking nominal trajectories, the system becomes unsafe, going as far as 32 m into the fire.

In FASTER, the same double integrator model is assumed, and a similar MPC problem is solved. We impose additional safety constraints, that the committed trajectory must lie within a safe flight corridor [107] based on the signed distance field to the fire, corrected based on the maximum fire spread rate. While this approach does keep the helicopter outside the fire, it gets surrounded by the fire (Fig. 3.7a). This is ultimately due to the fact that FASTER only plans trajectories over a finite planning horizon, and is therefore unable to guarantee recursive feasibility in a dynamic environment. Due to the large number of additional constraints on the QP, FASTER is about 3 times slower than the nominal planner.

In the Backup Filters approach, the backup trajectory is numerically forward propagated on the nonlinear system over the same 120 second horizon, and can be computed efficiently,

requiring less than 1 ms per iteration. Although this keeps the system safe, it does so at the cost of performance: the mean distance to the fire is 0.24 km, more than twice the target value, and the average speed is 10 m/s, 33% less than the target. This is because the desired flight direction is perpendicular to the backup flight direction, and therefore the executed trajectory is always off-nominal.

In `gatekeeper`, the committed trajectories are constructed by maximizing the interval that the nominal trajectory is tracked, before implementing the backup controller. This allows the system to follow the nominal, and deviate only when required to ensure safety. As before, the nominal trajectory is 120 s long, and the backup is simulated for $T_B = 120$ s. To initialize `gatekeeper`, the first candidate trajectory is constructed using just the backup controller, effectively setting $T_s = 0$. In our experiments this was sufficient ensure an initial valid committed trajectory.

In Fig. 3.7c, we see that `gatekeeper` chooses to not fly into the pocket, since it cannot ensure a safe path out of the pocket exists. `gatekeeper` is computationally lightweight, with a median run time of 3.4 ms, more than 20 times faster than FASTER. This is because `gatekeeper` searches over a scalar variable in a bounded interval, instead of optimizing $\mathbb{R}^{4N+2N-2}$ variables as in the MPC problem.

We studied the effect of conservatism in the environment model. For instance, suppose we assumed the max fire spread rate was 16 km/hr instead of 8 km/hr. Simulations showed that `gatekeeper` still maintains safe, but the resulting trajectories are more conservative: the mean distance to the fire increases by 37% to 0.148 km, and the mean speed decreases by 19% to 12.14 m/s. Despite doubling the level of conservatism in the environment model, we see a modest impact on the conservativeness of the resulting trajectories.

3.2.5.2 Quadrotor Navigation (Simulations)

We demonstrate the efficacy of the `gatekeeper` algorithm for a quadrotor flying through a previously unobserved area, in both a high fidelity simulation, and hardware experiments. The desired goal location is specified by the human operator. The quadrotor must simultaneously sense the environment, build a local map of the obstacles, plan a path to the goal, filter the path using `gatekeeper`, and finally execute the committed trajectory. All of the processing happens onboard, in realtime, and by using `gatekeeper`, the quadrotor does not crash into any obstacles. Each step of the perception-planning-control stack is described next, followed by a comparison to state-of-the-art methods. All simulations were run using Gazebo and RotorS [65], on an AMD Ryzen 7 5800h CPU 16 GB with a NVIDIA RTX 3050Ti. All hardware experiments were performed on a 16 GB Nvidia Xavier NX.

An environment with a forest of cylinders of random sizes, locations, and heights is

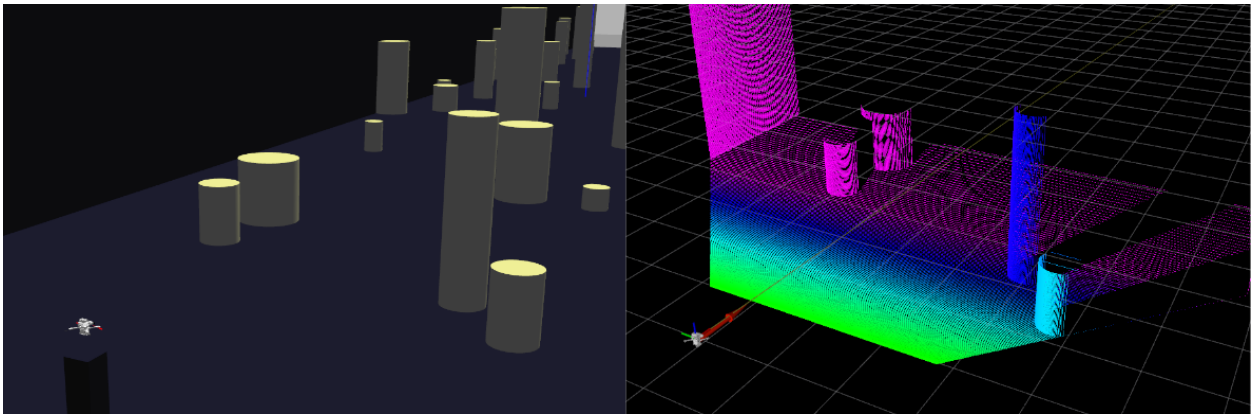

Figure 3.8: (Left) Simulation environment comprising of a quadrotor navigating in a 50 m long corridor with randomly scattered cylindrical obstacles of various heights and radii. This picture depicts the "Easy 1" world. (Right) The point-cloud sensor data received by the quadrotor describing the environment. Using the point-cloud, a SDF representation of the environment is constructed. A SFC, i.e., a convex polyhedron of obstacle-free space, centered on the quadrotor is extracted and used as the perceived safe set. The nominal planner treats unknown regions as free, while `gatekeeper` treats unknown regions as occupied.

generated in a corridor 50 m long, and 10 m wide. The start and goal locations are free, but a safe trajectory between these may not exist in environments with many obstacles. Since DMP is complete, the quadrotor will continue to explore until it finds a path to the goal.

Perception

The quadrotor is equipped with a front-facing Intel Realsense D455 camera, which has a limited field of view of $87° \times 58°$, and a limited sensing range of 8 meters, operating at 30 FPS. The incoming depth maps are fused into a ESDF representation using the NvBlox package [118] at a resolution of 7.5 cm.

Path Planner

From the ESDF, a 2D slice of the obstacle geometry between 0.8-1.2 m height is extracted. The path to the goal location is planned using the DMP [107], a computationally efficient alternative to A* which also pushes the path away from the obstacles. Unknown cells are treated as free cells. The planner takes less than 30 ms to replan trajectories, and is operated at 5 Hz. Given a desired linear travel speed $v = 2$ m/s, time is allocated to each leg of the returned path to construct the trajectory. This trajectory is not dynamically feasible for the quadrotor's nonlinear dynamics.

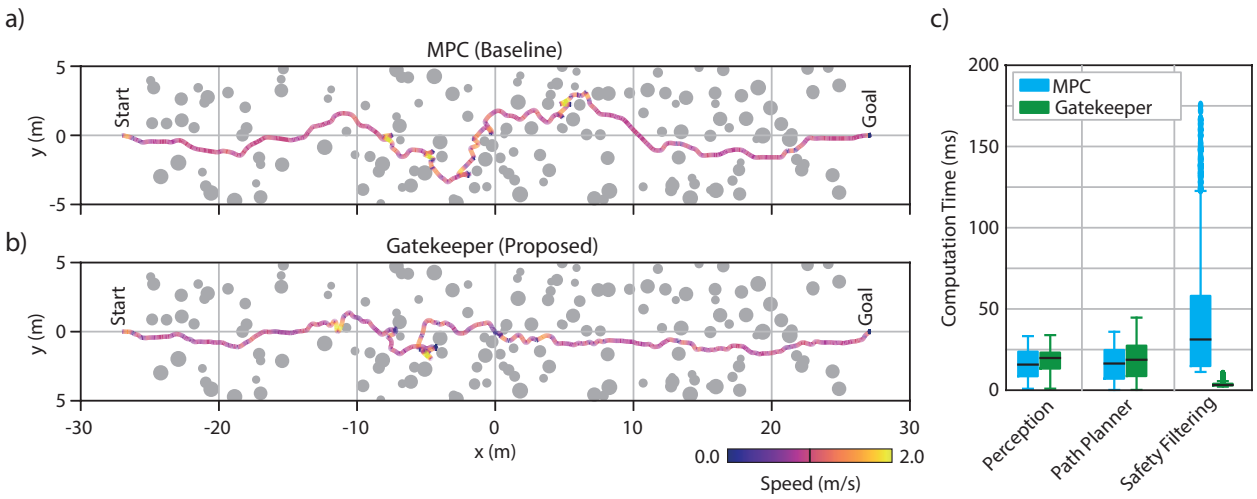

Figure 3.9: Trajectories executed in the "Hard 1" world, using (a) MPC-based safety filter (baseline) and (b) the proposed `gatekeeper`-based safety filter. The gray circles indicate obstalces. Visually, the paths are similar, and are traversed with similar speeds. The color indicates that for most of the trajectories, the speed is at the target of 1 m/s, but near the obstacles (where there is greater replanning), the speeds vary more. (c) Box-and-whiskers plot showing the computation time for perception, planning, and safety filtering. The computation time for perception and path planning is similar with both safety filters, since both use the same perception and path planning implementation. However, `gatekeeper` is significantly faster than the MPC-based safety filter.

Safety Filtering

In our implementation of `gatekeeper`, a $4 \times 4 \times 2$ meter block centered on the quadrotor is extracted from the ESDF. unknown voxels are treated as obstacles. A convex polyhedron representing the safe region is constructed using the DecompUtil [107], and is the safe set $\mathcal{B}_k$ used in `gatekeeper`. The environment is assumed static, but is unknown at the start of the run - as new regions are observed, the perceived safe set expands to include new regions.

Next, `gatekeeper` (as described in Algorithm 3.1) is used to convert the nominal trajectory into a dynamically feasible and safe trajectory for the quadrotor to follow.[7] The backup controller used is a stopping controller. For p^{can} to be valid, it must (A) lie within the safe polyhedron mentioned above (accounting for the grid resolution (0.075 m), the quadrotor's radius (0.15 m), and a robustness margin $R = 0.1$ m)[8], (B) terminate within the safe polyhedron accounting for the quadrotor radius and a robustness margin $R + r = 0.2$ m, (C) terminate with zero speed and zero control input. These conditions guarantee that the

[7]We used a triple integrator model to validate trajectories, as in [110, 159, 164]. We tried the nonlinear model in [100], but the communication latency between the Pix32 and Xavier NX degraded performance.

[8]Robustness margins were determined by flying the quadrotor, and measuring the tracking error as it executed some trajectories. The measured error was 5-10 cm, and thus $R = 0.1$ m was chosen.

Table 3.4: Summary of simulations in 15 different worlds with 3 difficulty levels, comparing the performance of an MPC-based safety filter against `gatekeeper`. `gatekeeper` is able to successfully reach the goal in more scenarios, and is an order of magnitude computationally faster.

	Goal Reached?		Median Comp. Time [ms]		Max Comp. Time [ms]		Average Speed [m/s]	
World	MPC	GK	MPC	GK	MPC	GK	MPC	GK
easy 1	True	True	34.71 ± 0.10	3.28 ± 0.04	168.89	10.78	0.91	0.81
easy 2	True	True	35.55 ± 0.11	3.42 ± 0.05	161.59	10.51	0.82	0.77
easy 3	True	True	33.18 ± 0.12	3.26 ± 0.05	151.12	12.48	0.83	0.67
easy 4	True	True	34.17 ± 0.25	3.33 ± 0.05	172.15	11.70	0.83	0.45
easy 5	True	True	35.40 ± 0.20	3.36 ± 0.05	180.94	11.94	0.90	0.35
medium 1	True	True	39.78 ± 0.20	3.27 ± 0.06	178.21	10.62	0.78	0.61
medium 2	False	True	47.62 ± 1.10	3.27 ± 0.05	217.97	12.19	0.57	0.76
medium 3	True	True	33.65 ± 0.08	3.18 ± 0.04	199.93	11.53	0.79	0.75
medium 4	False	False	44.93 ± 2.03	3.23 ± 0.06	199.74	9.25	0.44	0.43
medium 5	True	True	27.72 ± 0.10	3.23 ± 0.05	211.08	10.82	0.81	0.82
hard 1	True	True	31.22 ± 0.16	3.18 ± 0.07	201.54	9.65	0.68	0.82
hard 2	False	True	56.61 ± 0.61	3.41 ± 0.08	184.23	12.06	0.68	0.79
hard 3	False	False	44.75 ± 0.54	3.35 ± 0.06	213.06	9.43	0.34	0.54
hard 4	True	True	13.60 ± 0.09	3.25 ± 0.04	218.98	10.41	0.34	0.73
hard 5	False	True	56.57 ± 4.26	3.25 ± 0.06	208.30	10.15	0.50	0.68

quadrotor can hover indefinitely at the terminal position.[9] The maximum switch time was $T_S \leq 2$ s, and the backup trajectory was propagated from $T_b = 2.0$ s. Note, due to the safety filtering, the nominal trajectory may not be traversable, for instance through a narrow passage. When this occurs, `gatekeeper` publishes a virtual obstacle, forcing the nominal planner to replan alternative routes.

Tracking Controller

The last committed trajectory tracked using a geometric tracking controller [100], running at 250 Hz.

[9]Here, we do not consider the quadrotor's limited battery life as a constraint. This is addressed in [126] using the `gatekeeper` strategy.

Benchmark

Our implementation of `gatekeeper` is compared with an MPC safety filter. In the MPC filter, the following optimization problem is solved:

$$\begin{aligned} \underset{x\in\mathcal{X}^{N+1},u\in\mathcal{U}^N}{\operatorname{argmin}} & \sum_{i=0}^{N} \|x_i - [p_k^{nom}]_i\|_Q^2 + \sum_{i=0}^{N-1} \|u_i - [u_k^{nom}]_i\|_R^2 \\ \text{s.t. } & x_{i+1} = Ax_i + Bu_i \\ & x_i \in \mathcal{B}_k \\ & x_0 = \hat{x}(t_k), \quad x_N = x_{N-1} \end{aligned}$$

where $[p_k^{nom}]_i = p_k^{nom}(t_k + i\Delta T)$ where $\Delta T = 0.02$ seconds is the discretization step size. A planning horizon of 2 seconds is considered, same as in our `gatekeeper` implementation. $[u_k^{nom}]_i$ the corresponding control input. To avoid solving a nonlinear program, the dynamics model assumed for the MPC safety filter is the linear double integrator. The set $\mathcal{B}_k$ is the same safe flight polyhedron used in `gatekeeper`. The initial condition is required to match with the estimated state, and the last constraint ensures that the quadrotor trajectory terminates within the horizon. The resulting problem is a quadratic program, and solved using OSQP [157].

Results

Fifteen different world environments were constructed with 3 difficulty levels, defined by the density of obstacle cylinders (Fig. 3.8). The quadrotors were tasked to fly a linear distance of 54 meters, at a desired speed of 1 m/s. Figure 3.9 shows sample trajectories and compares the computational cost of the baseline (MPC safety filter) and proposed (gatekeeper) strategies. Table 3.4 summarizes the performance of each safety filter. Both the MPC and `gatekeeper` algorithms prevented collisions. In some cases neither MPC nor `gatekeeper` were able to reach the goal location, although `gatekeeper` was able to find a trajectory in more cases than MPC. MPC was consistently slower than the `gatekeeper`, requiring approximately 10x the computation time. Finally the average speed of the quadrotor was similar with both filters. Through this, we conclude that the performance of `gatekeeper` and MPC are similar, although `gatekeeper` computationally efficient, while additionally handling the nonlinear dynamics of the quadrotor.

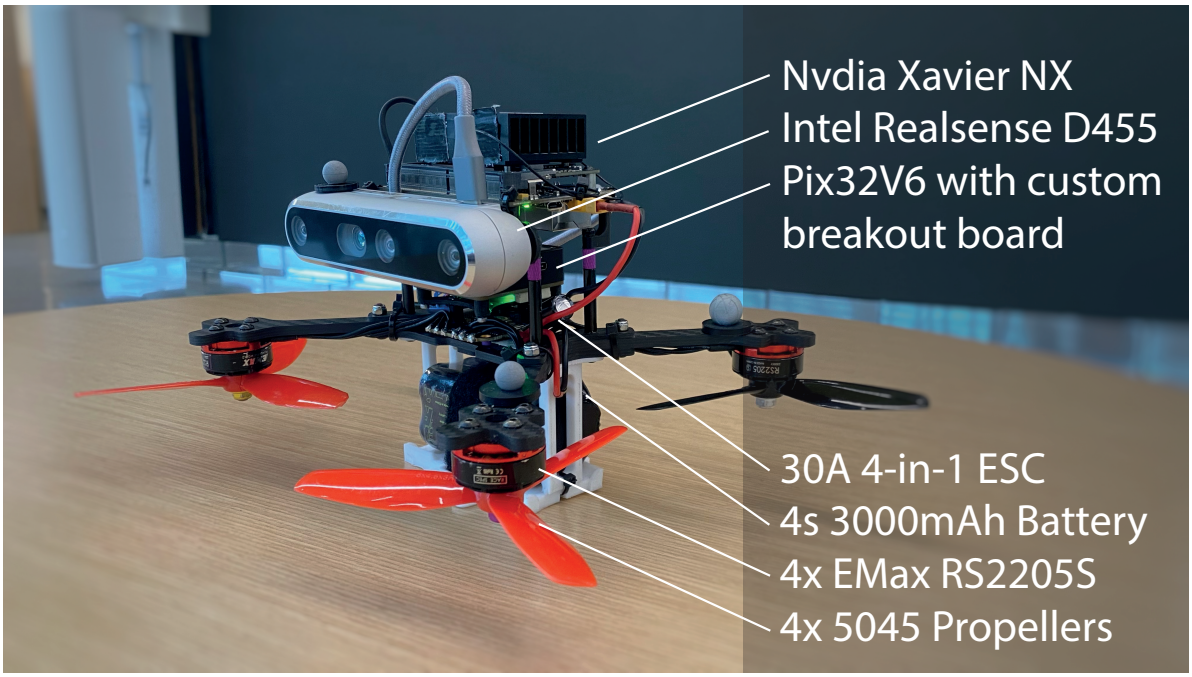

Figure 3.10: Quadrotor used for experiments. A combination of off-the-shelf components and custom breakout boards is used to minimize weight and maximize performance.

3.2.5.3 Quadrotor Navigation (Experiments)

We also demonstrate the algorithm experimentally. A custom quadrotor was designed to optimize the payload, and maximize the flight time (Fig. 3.10). The quadrotor's wet weight is 820 g, with a 15 min hover flight time. The perception, planning, and safety filtering steps were all performed on an onboard computer, the NVIDIA Xavier NX. The low-level geometric controller [100] was implemented on a Pix32V6c, communicating with the Xavier over UART. The goal destination and yaw angle was specified by a human operator.

As in the simulations, the quadrotor uses the Realsense D455 camera's RGBD images to construct a map of the environment using NvBlox, plans a trajectory using DMP, and filters the trajectory using `gatekeeper`.[10] The last committed trajectory is tracked using the geometric controller. Each of these steps were implemented as described in Section 3.2.5.2.

Figure 3.11 shows top-down snapshots of the map, and both the nominal and committed trajectories. Initially, the quadrotor tries to fly through a gap between the green and red obstacles. However, since the gap is too small for `gatekeeper` to certify that it is safe to traverse through the gap,[11] new trajectories are not committed, and the quadrotor executes

[10] In hardware, the path planner replans once every 2 seconds. The ESDF is updated at 5 Hz, and `gatekeeper` is run at 20 Hz.

[11] The minimum gap required is the sum of (quadrotor diameter, 0.3 m) + (voxel size of map, 0.075 m) + (tracking radius r, 0.1 m) + (robustness radius R 0.1 m) = 0.58 m. The gap was measured to be 0.45 m

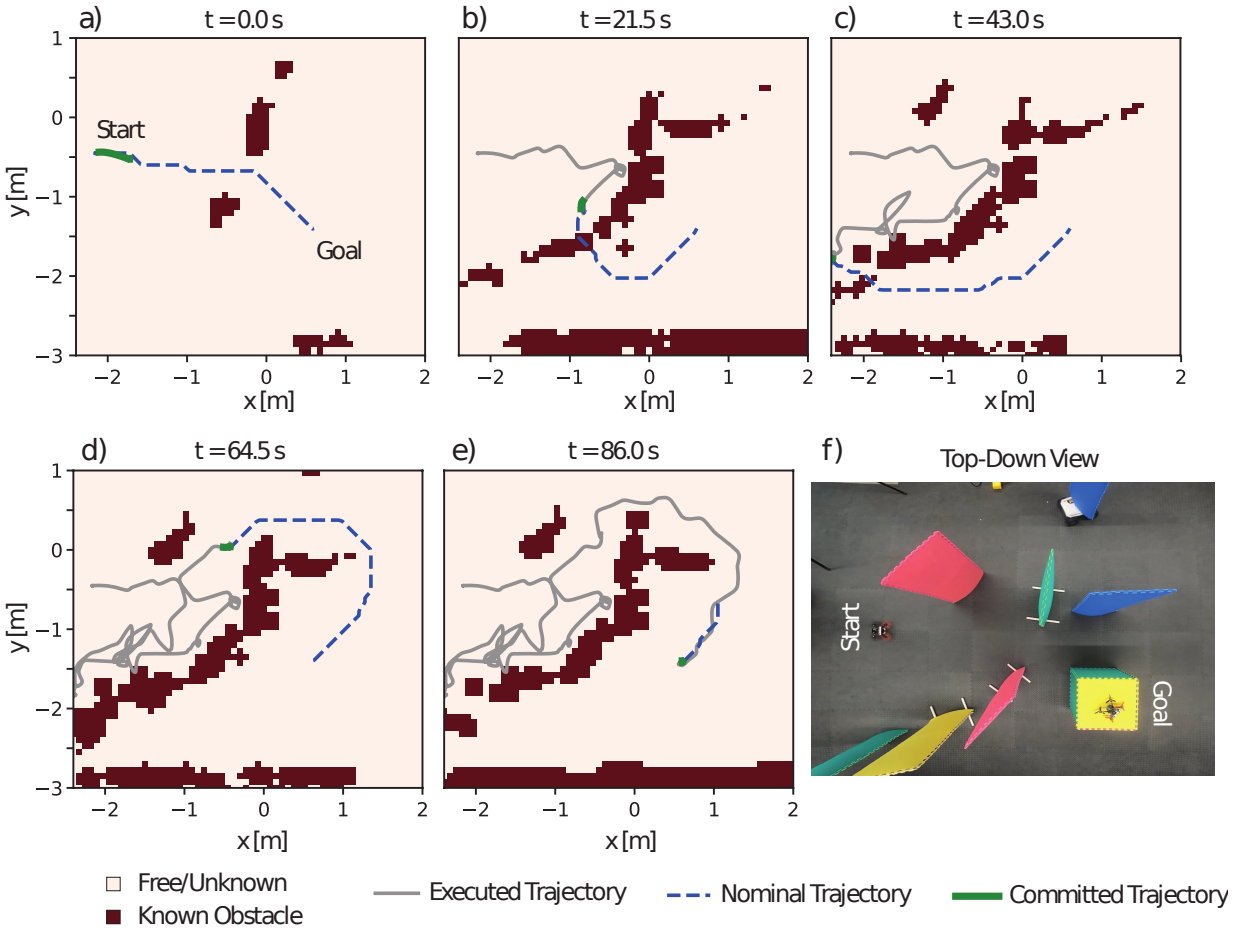

Figure 3.11: (a-e) Snapshots of the map, nominal trajectory, committed trajectory, and executed path of the quadrotor. (f) Top-down view of the obstacle geometry.

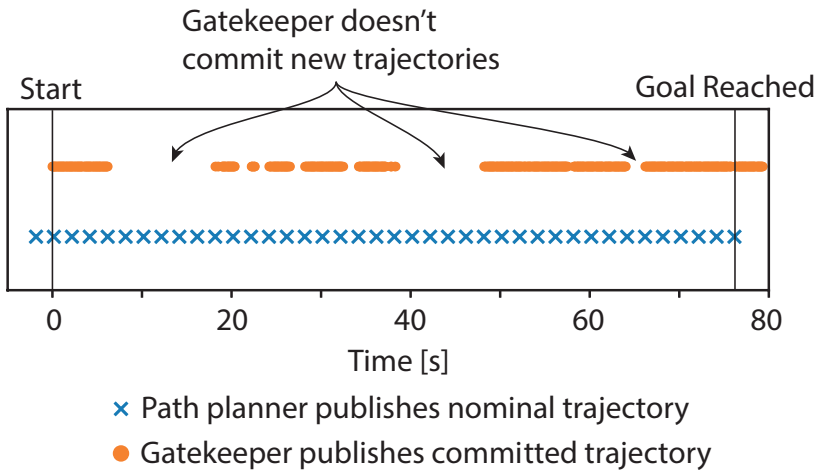

Figure 3.12: Each dot (cross) represents a timepoint when the committed (nominal) trajectory is published. The gaps represent intervals when `gatekeeper` prevents unsafe trajectories from being committed. During these times, the controller continues to track the the last committed trajectory.

its backup controller: stop and yaw. Once the nominal planner plans a new trajectory towards the right `gatekeeper` allows a new trajectory to be committed. However, as the quadrotor approaches this gap, again it is too narrow to safely traverse. This repeats a few times before eventually the nominal planner finds the trajectory that indeed is safe to traverse, and the quadrotor reaches the goal destination.

In Fig. 3.12, the times at which nominal and committed trajectories are published are plotted. In our implementation using ROS2, when no new candidate trajectories are valid (as in (3.55)), the `gatekeeper` node does not publish a new committed trajectory, and therefore the controller continues to track the last committed trajectory. Therefore, in Fig. 3.12, the path-planner publishes at regular intervals, but there are gaps when `gatekeeper` is running but not publishing new committed trajectories. To allow the system to continue making progress towards the goal, we publish a virtual obstacle along the nominal trajectory when this happens, forcing the path planner to find a new trajectory to the goal. In this particular run, we observed 9 such instances.

The experiments also highlighted some limitations of `gatekeeper` that can form the basis for future study. In particular, suppose a nominal planner is poorly designed, and produces trajectories that are collision free, but not desirable, e.g., if the nominal plan causes the drone to jerk back and forth and yaw rapidly. Such a nominal trajectory could pass the validity check 3.13, but could lead to the quadrotor chattering. In the future, we wish to investigate how to co-design the `gatekeeper` with planners and controllers to avoid such undesired

across.

trajectories. In our experiments, we have sometimes observed the nominal planner making large and abrupt changes in the nominal trajectory, but the quadrotor was able to track the committed trajectories.

Finally, further investigation into the design of backup controllers could yield interesting directions for future research. In our current implementation, the backup controller stops the quadrotor along the nominal trajectory. However, the position at which the quadrotor comes to a stop could be designed, for example, to maximize the visibility of the unknown regions. Such a backup controller would still maintain safety but also allow the quadrotor to reason more efficiently about the environment it is operating in. Furthermore, to operate this quadrotor in an environment with dynamic obstacles further attention will be needed on the design of backup controllers. If one were to assume a bounded speed at which the environment could move, the safe flight polyhedrons could quickly collapse to becoming empty. Using semantic maps, for example [140], might help to identify the dynamic parts of the environment, and overcome this issue.

3.2.6 Conclusion

This section proposes an algorithm ("`gatekeeper`") to safely control nonlinear robotic systems while information about dynamically-evolving safe states is received online. The algorithm constructs an infinite-horizon committed trajectory from a nominal trajectory using backup controllers. By extending a section of the nominal trajectory with the backup controller, `gatekeeper` is able to follow nominal trajectories closely, while guaranteeing a safe control input is known at all times. We have implemented the algorithm in a simulated aerial firefighting mission and on-board a real quadrotor, where we demonstrated `gatekeeper` is less conservative than similar methods, while remaining computationally lightweight. Various comparisons to state-of-the-art techniques are also provided.

A key benefit of the `gatekeeper` approach is its applicability in dynamic environments where the safe set is sensed online. This allows the method to be applied to a wide range of scenarios where only limited safety information is known, for instance, overtaking and merging scenarios for autonomous vehicles. A limitation of `gatekeeper` is the difficulty in finding backup controllers and sets that are suitable for the robotic system and environment considered, particularly in time-varying environments. Ultimately the possible safety guarantees rely on the ability to make forecasts of the environment given limited sensing information. Furthermore, when there are multiple safety conditions that must all be satisfied simultaneously, one could either design a single backup controller to satisfy all the constraints, or design multiple separate backup controllers and switch between them. Both of

these approaches have their challenges and their suitability is case-dependent. These strategies require further analysis, an interesting direction for future work. Future directions also include developing more general methods to identify backup controllers, and understanding how the method can be applied in adversarial multi-agent settings.

3.2.7 Appendix: Worked Example for the Firewatch Scenario

This example demonstrates how the sets $\mathcal{S}(t), \mathcal{B}_k(t), \mathcal{C}_k(t)$ are related, using the firewatch mission. For simplicity, consider a double integrator system,

$$\dot{x} = Ax + Bu \tag{3.67}$$

where $x_{pos} = [x_1, x_2]^T$ is the position of the helicopter, and $x_{vel} = [x_3, x_4]^T$ is the velocity.

Say the fire starts at $t = t_0$, at location $p = [0, 0]^T$. The fire expands radially, with rate of spread $\sigma : \mathbb{R}^2 \to \mathbb{R}_{\geq 0}$, i.e., $\sigma(p)$ is the rate of spread at a location $p \in \mathbb{R}^2$. To simplify the algebra, assume the RoS depends only on $\|p\|$, i.e, $\sigma(p_1) = \sigma(p_2)$ for any $\|p_1\| = \|p_2\|$. This means that the fire always spreads out uniformly in a circle.

Therefore, the safe set is time-varying, described by

$$\mathcal{S}(t) = \left\{ x : \|x_{pos}\| \geq \int_0^t \sigma(r(\tau))d\tau \right\} \ \forall t \geq 0 \tag{3.68}$$

where $r(t)$ is the radius of the fire at time $t \geq t_0$.

Since we don't know σ, we don't know $\mathcal{S}(t)$. Instead, we assume a reasonable upper bound: $\sigma(r) \leq 2$ m/s for all $r \geq 0$.

Therefore, at $t = t_0$, we can define an *perceived safe set*:

$$\mathcal{B}^0(t) = \{x : \|x_{pos}\| \geq 2(t - t_0)\} \ \forall t \geq t_0 \tag{3.69}$$

and clearly $\mathcal{B}^0(t) \subset \mathcal{S}(t) \ \forall t \geq 0$. Notice that $\mathcal{B}^0(t)$ is *not* a controlled invariant set for the double integrator.[12]

Suppose the system can directly measure the fire's radius. Let the k-th measurement be $r_k = r(t_k)$. This allows us to define the k-th *perceived* safe set:

$$\mathcal{B}_k(t) = \{x : \|x_{pos}\| \geq r_k + 2(t - t_k)\} \ \forall t \geq t_k \tag{3.70}$$

[12]Technically, a higher-order CBF could be used to design a QP controller that renders a subset of $\mathcal{B}^0(t)$ forward invariant, but this is only possible since $\mathcal{B}^0(t)$ is a sufficiently smooth function that we can analyze analytically.

One can verify

$$\begin{aligned}
\mathcal{B}_k(t) &= \{x : \|x_{pos}\| \geq r_k + 2(t - t_k)\} \\
&= \left\{x : \|x_{pos}\| \geq \int_{t_0}^{t_k} \sigma(r(\tau))d\tau + 2(t - t_k)\right\} \\
&\subset \left\{x : \|x_{pos}\| \geq \int_{t_0}^{t_k} \sigma(r(\tau))d\tau + \int_{t_k}^{t} \sigma(r(\tau))d\tau\right\} \\
&= \left\{x : \|x_{pos}\| \geq \int_{t_0}^{t} \sigma(r(\tau))d\tau\right\} \\
&= \mathcal{S}(t)
\end{aligned}$$

i.e. $\mathcal{B}_k(t) \subset \mathcal{S}(t)$ for all $t \geq t_k$.

Similarly, we can verify that for any $k \geq 0$,

$$\begin{aligned}
\mathcal{B}_{k+1}(t) &= \{x : \|x_{pos}\| \geq r_{k+1} + 2(t - t_{k+1})\} \\
&= \left\{x : \|x_{pos}\| \geq r_k + \int_{t_k}^{t_{k+1}} \sigma(r(\tau))d\tau + 2(t - t_{k+1})\right\} \\
&\supset \{x : \|x_{pos}\| \geq r_k + 2(t_{k+1} - t_k) + 2(t - t_{k+1})\} \\
&= \{x : \|x_{pos}\| \geq r_k + 2(t - t_k)\} \\
&= \mathcal{B}_k(t)
\end{aligned}$$

i.e., $\mathcal{B}_k(t) \subset \mathcal{B}_{k+1}(t)$ for all $t \geq t_k$.

This proves that Assumption 3.4 is satisfied. Next, we define the backup controllers.

For any $k \in \mathbb{N}$, suppose the state is $x_k = x(t_k)$. The backup controller should drive the system radially away from the fire. Define n_k as the unit vector pointed at $x(t_k)$:

$$n_k = x_{pos}(t_k) / \|x_{pos}(t_k)\| \tag{3.71}$$

Notice that if the position followed the reference

$$p_{ref}(t) = (1 + r_k + 2(t - t_k))n_k \tag{3.72}$$

then the reference is moving radially at a speed of 2 m/s, and therefore faster than the maximum spread rate of the fire. Thus $p_{ref}(t)$ is a safe trajectory for all $t \geq t_k$.

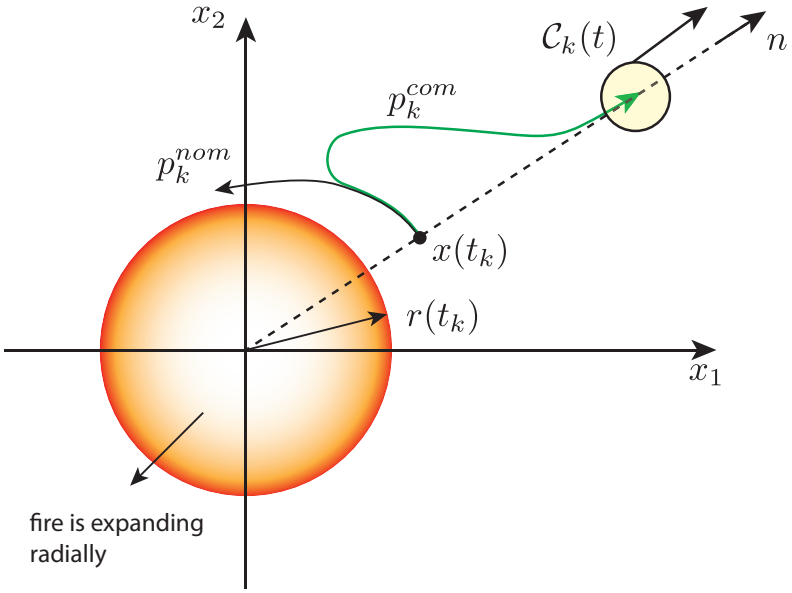

Figure 3.13: Depiction of the scenario in the worked example.

This leads to the following backup controller:

$$\pi_k^B(t,x) = -K\left(\begin{bmatrix} x_{pos} \\ x_{vel} \end{bmatrix} - \begin{bmatrix} p_{ref}(t) \\ 2n_k \end{bmatrix}\right) \tag{3.73}$$

where $K \in \mathbb{R}^{2\times 4}$ is a stabilizing LQR gain for the double integrator. This controller stabilizes the system to $\mathcal{C}_k(t)$, where

$$\mathcal{C}_k(t) = \left\{ x : \left\| x - \begin{bmatrix} p_{ref}(t) \\ 2n_k \end{bmatrix} \right\| \leq 1 \right\} \tag{3.74}$$

This set is controlled invariant using the backup controller π_k^B. Geometrically, $\mathcal{C}_k(t)$ is a unit norm ball that is moving radially at 2 m/s in the n_k direction. Therefore, $\mathcal{C}_k(t) \subset \mathcal{S}(t)$ for all $t \geq t_k$, since the set is moving outwards radially at a speed higher than the maximum spread rate.

This example demonstrates how $\mathcal{S}(t)$, $\mathcal{B}_k(t)$, $\mathcal{C}_k(t)$ can be defined for a given problem. The main validation step in `gatekeeper`, will confirm whether after following the nominal trajectory over $[t_k, t_k+T_S)$, the system is able to safely reach $\mathcal{C}_k(t)$ using the backup controller π_k^B.

While the sets were described analytically here, in simulations they were represented numerically using SDFs.

CHAPTER 4

Safety-Critical Perception

In the previous chapter, we have developed two methods of safety filtering at the planning level. However, both assumed that the perception outputs are to some extent perfect. For instance, in the differential flatness strategy we assumed both the state estimate and the safe sets were known apriori exactly. In the `gatekeeper` strategy, we allowed for some uncertainty in the state estimate, and in the dynamic evolution of the safe set, but we still assumed that the safe set is known.

Here we start to relax this assumption. Our objective is to build autonomous systems that can operate entirely using onboard sensors, and as part of this the perception modules must produce both the state estimate as well as the obstacle map. This is often referred to as the Simultaneous Localization and Mapping (SLAM) problem.

However, classical perception literature on SLAM focuses primarily on the accuracy of the methods, instead of the correctness: there is often only empirical evidence that the algorithm will correctly estimate the state estimate and the locations of the obstacles, and often do not provide any indication of the maximum error of the estimates in real-time.

Here, we propose a modification to the obstacle mapping methods such that such bounds can be derived, and identify a desired structure for these bounds such that they can integrate with the rest of the planning and control modules effectively.

4.1 Certifiably Correct Obstacle Mapping Despite Odometry Drift

Accurate state estimation and mapping are essential for safe robotic navigation, as planners and controllers rely on perception outputs to ensure the safety of planned trajectories or control actions. Various methods have been developed to certify that controllers meet predefined safety specifications [21, 69], and when real-time obstacle detection is necessary, it is often intuitive to handle safety constraints in the planner [10, 110, 164]. These methods typically assume perfect perception, a simplification that can lead to safety violations.

A perception module provides a pose estimates and constructs maps that represent obstacle geometry, and can take a variety of formats, such as ESDFs [118, 128], polytopic SFCs [107], occupancy log-odds [80], or NERFs [141]. Although recent advances in perception algorithms have achieved significant accuracy improvements [45, 117, 147, 161, 182], formal error analysis is often lacking. Without quantified error bounds, guaranteeing the safety of a closed-loop robotic system remains a challenge.

This section introduces a framework for "certifiably correct mapping" ensuring that obstacle-free regions of a map remain correct despite odometry drift. The challenge is illustrated in Figure 4.1. Consider an environment $\mathcal{W} = \mathcal{F} \cup \mathcal{O}$, representing free and obstacle spaces, respectively (Figure 4.1a). As a robot navigates, at the k-th time step it has created a map $\mathcal{M}_k$, comprising the supposedly safe space $\mathcal{S}_k$, the unknown space $\mathcal{U}_k$ and the recognized obstacles $\mathcal{R}_k$ (Figure 4.1b). However, due to odometry drift, maps can misclassify obstacles as free space, leading to potential safety violations as indicated in Figure 4.1c. We address this by deflating safe regions in order to ensure $\mathcal{S}_k \subset \mathcal{F}$ at all times (Figure 4.1d).

Recent work has explored perception with correctness guarantees. For example, [139] achieves global optimization in pose graph optimization problems via a convex reformulation, and [113] offers error-bounded localization in convex environments. The methods in [3, 179] propose certifiably correct pointcloud registration and visual odometry. Similarly [183] demonstrated that bounded attitude errors lead to bounded position errors. Compared to [3] this section assumes that the incremental pose estimate is bounded in a Lie-algebraic sense, allowing the proposed methods to be used with a wider range of odometry algorithms than one in [3].

Our main contributions are as follows:

- The theoretical framework to construct and deflate the free space in obstacle maps to ensure their correctness despite odometry drift. Assuming the odometry algorithm reports the pose and the covariance of the incremental transform, we propose deflating

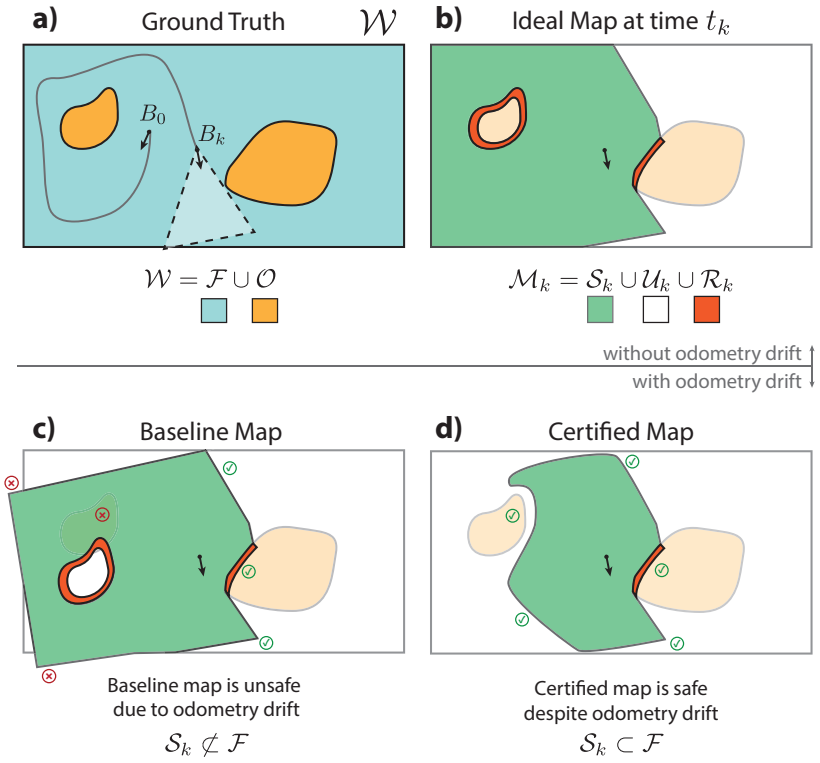

Figure 4.1: Overview of notation and objectives. (a) depicts the operating environment, where the world $\mathcal{W}$ is the union of the free space $\mathcal{F}$ and the obstacles $\mathcal{O}$. The robot does not know $\mathcal{F}$ or $\mathcal{O}$. It starts at B_0, and follows the gray trajectory to B_k building the map as it goes. (b) depicts the ideal mapping output, where at the k-th timestep, the map $\mathcal{M}_k$ is composed of the known safe region $\mathcal{S}_k$, the unknown space $\mathcal{U}_k$ and the known obstacle set $\mathcal{R}_k$. (c) depicts the map produced by current state-of-the-art methods, where due to odometry drift the map is erroneous: notice that the safe region (according to the constructed map) is not a subset of the free space, $\mathcal{S}_k \not\subset \mathcal{F}$. (d) depicts the desired behavior of the certified maps, where although the safe region is smaller, it is certifiably-correct: we can prove that $\mathcal{S}_k \subset \mathcal{F}$.

the supposedly safe region ($\mathcal{S}_{k+1}$ is deflated relative to $\mathcal{S}_k$) to ensure that it remains a subset of the free region $\mathcal{F}$.

- We prove the correctness and applicability of this framework on two popular and state-of-the-art mapping frameworks: the polytopic SFCs of [107] and the ESDFs of [118].
- Beyond providing the theoretical analysis and proofs of correctness, we validate and compare our approach with state-of-the-art baseline methods through extensive simulations on the Replica dataset [158].
- Finally, we demonstrate the approach in a real-world experiment on a robotic rover. A human teleoperates the rover using only the FPV feed and the obstacle map constructed and streamed to the operator in real-time. The rover uses an onboard safety filter to prevent collisions. Unlike baseline methods which result in collisions, our approach prevents crashes by deflating the safe regions appropriately.

It is critical that we deflate $\mathcal{S}_k$ rather than inflate known obstacles $\mathcal{R}_k$. If the obstacles are inflated based on the accumulated odometry error, these obstacles can only grow in size, and might eventually occupy the entire domain $\mathcal{W}$. Instead, by deflating a safe region $\mathcal{S}_k$, the region that is certifiably safe shrinks, eventually becomes an empty set, and is removed from memory (i.e., the region becomes part of $\mathcal{U}_k$). When the region is observed by a sensor again, it can again be added to $\mathcal{S}_k$ again. Computationally, this reduces memory requirements, and mathematically this allows us to treat deflated obstacles as unknown regions and plan paths accordingly.

4.1.1 Preliminaries and Problem Statement

Matrix Lie Groups

Here we present a brief review of Matrix Lie Groups in the context of this section, with additional equations and details in Section 4.1.8.1. We refer the reader to the excellent references [26, 111, 155] for a more complete description.

The Lie group $\mathbb{SO}(3)$ defines 3D rotations, and the group $\mathbb{SE}(3)$ defines 3D rigid transformations. Both $\mathbb{SO}(3)$ and $\mathbb{SE}(3)$ are Matrix Lie groups, i.e., group elements are matrices, and composition operator is the standard matrix multiplication operator. In $\mathbb{SE}(3)$ the group action $\cdot : \mathbb{SE}(3) \times \mathbb{R}^3 \to \mathbb{R}^3$ transforms a point p from its representation in frame A to that

in frame B. Given $T_A^B = \begin{bmatrix} R & t \\ 0 & 1 \end{bmatrix} \in \mathbb{SE}(3)$,

$$p|^B = T_A^B \cdot p|^A = Rp|^A + t. \tag{4.1}$$

The Lie algebra of a group is a vector space of all possible directions a group element can be perturbed locally. The Lie algebras of $\mathbb{SO}(3)$ and $\mathbb{SE}(3)$ are $\mathfrak{so}(3)$ and $\mathfrak{se}(3)$ respectively. These vector spaces are isomorphic to $\mathbb{R}^3$ and $\mathbb{R}^6$ respectively. The $\wedge$ operator converts the Euclidean vector to an element of the Lie Algebra, and $\vee$ does the inverse.

Consider a Lie group $\mathbb{G}$ with an associated Lie algebra $\mathfrak{g}$ that is isomorphic to the Euclidean vector space $\mathbb{R}^n$. Given an element $x \in \mathfrak{g}$, we can convert it to the corresponding group element using the exponential map, $\exp : \mathfrak{g} \to \mathbb{G}$,

$$\exp(x) = \sum_{k=0}^{\infty} \frac{x^k}{k!} = I + x + \frac{x^2}{2} + \cdots \tag{4.2}$$

where I is the identity element in $\mathbb{G}$. For convenience, we also define the Exp map, which maps from the Euclidean vector space to the group directly, $\operatorname{Exp} : \mathbb{R}^n \to \mathbb{G}$,

$$\operatorname{Exp}(\xi) = \exp(\xi^\wedge). \tag{4.3}$$

The corresponding inverse operations are $\log : \mathbb{G} \to \mathfrak{g}$ and $\operatorname{Log} : \mathbb{G} \to \mathbb{R}^n$. For certain groups including $\mathbb{SE}(3)$, these operations have analytic expressions [155, Appendix].

Uncertain Poses and Transforms

An uncertain pose or transform $T_A^B \in \mathbb{SE}(3)$ is denoted

$$T_A^B \sim \mathcal{N}(\widehat{T}_A^B, \Sigma_T),$$

where $\widehat{T}_A^B \in \mathbb{SE}(3)$ is the mean estimate, and $\Sigma_T \in \mathbb{S}_{++}^6$ is a covariance matrix. This indicates T_A^B is the transform

$$T_A^B = \widehat{T}_A^B \operatorname{Exp} \tau, \tag{4.4}$$

where $\tau \in \mathbb{R}^6$ is a random sample drawn from $\tau \sim \mathcal{N}(0, \Sigma_T)$.

Recall the group action $p|^B = T_A^B \cdot p|^A$. If the transform T_A^B is uncertain, $p|^B$ follows a

distribution and, to first order, is a normal distribution [26, 155]:

$$p|^B = \left(T_A^B \cdot p|^A\right) \sim \mathcal{N}(\hat{p}|^B, \Sigma_p) \tag{4.5}$$

where the mean and covariance are

$$\hat{p}|^B = \widehat{T}_A^B \cdot p|^A \in \mathbb{R}^3, \quad \Sigma_p = J\Sigma_T J^T \in \mathbb{S}^3_{++}$$

with $J = \begin{bmatrix} R & -R[p|^A]_\times \end{bmatrix} \in \mathbb{R}^{3\times 6}$.

For the remainder of the section, we truncate the distribution making the following assumption:

Assumption 4.1. *Let $T_A^B \sim \mathcal{N}(\widehat{T}_A^B, \Sigma)$, where $\widehat{T}_A^B = \begin{bmatrix} R & t \\ 0 & 1 \end{bmatrix}$. Then for any $p|^A \in \mathbb{R}^3$, the point $p|^B \in \mathbb{R}^3$ satisfies*

$$p|^B = T_A^B \cdot p|^A \in \mathcal{E} \tag{4.6}$$

where $\mathcal{E} \subset \mathbb{R}^3$ is the ellipsoid

$$\mathcal{E} = \left\{ p \in \mathbb{R}^3 : \left\| \Sigma_p^{-1/2} \left(p - \widehat{T}_A^B \cdot p|^A \right) \right\| \leq 1 \right\}, \tag{4.7}$$

$$\Sigma_p = \kappa J \Sigma J^T \in \mathbb{S}^3_{++}, \quad J = \begin{bmatrix} R & -R[p|^A]_\times \end{bmatrix} \in \mathbb{R}^{3\times 6}.$$

for some $\kappa > 0$.[1]

In other words, the assumption is that when a point p is transformed from its representation in frame A to that in frame B, the point $p|^B$ is contained within an ellipsoid $\mathcal{E}$ centered on the estimated point $\widehat{T}_A^B \cdot p|^A$, as defined in (4.7). The size and principal axes of the ellipsoid are defined by the estimated transform $\widehat{T}_A^B$ and the covariance matrix Σ. This allows us to bound the error of mapping points between frames, and the bound can be made tighter if κ is increased, or if higher order approximations are used, as in [26]. Since we focus on rototranslations between successive body frames, the transforms T_A^B should be close to identity where the first order approximations work well.

[1] κ chooses the probability the bound contains the point. For a d-dimensional normal distribution, $x \sim \mathcal{N}(\mu, \Sigma)$, the probability that $\left\|(\kappa\Sigma)^{-1/2}(x-\mu)\right\| \leq 1$ is $p \in [0,1]$ such that $\kappa = \chi_d^2(p)$, where χ_d^2 is the quantile function of the chi-squared distribution with d degrees of freedom. For 3D points, $\kappa = 2$ corresponds to $p = 97\%$.

Reference Frames

This section uses the inertial frame I, a mapping frame M, and the body-fixed frame at the k-th timestep, B_k. Usually, M and I are equivalent, and M is defined such that at $M = B_0$. However, since we are considering odometry drift, M can drift relative to I. We assume that I is the true inertial frame (in which the obstacles are static), and M is the reference frame used to construct the state estimate and the map.

Problem Statement

Let $\mathcal{O}$ represents the obstacle geometry in a static environment $\mathcal{W} \subset \mathbb{R}^3$. Both $\mathcal{O}$ and $\mathcal{F} = \mathcal{W} \backslash \mathcal{O}$ are assumed initially unknown. We assume neither set contains any isolated points, and that $\mathcal{F}$ is closed. As with points, a set can be represented in a frame, i.e., we say that $\mathcal{O}|^{B_k} \subset \mathbb{R}^3$ is the set of all obstacle points represented in frame B_k.

To avoid obstacles, we must build a map of the environment. At the k-th timestep, the map is $\mathcal{M}_k$, which consists of the (claimed) free-space $\mathcal{S}_k$, the unknown space $\mathcal{U}_k$, and the (claimed) obstacle space $\mathcal{R}_k$. We say that a map is correct if the claimed free space is indeed a subset of the true free space. More formally,

Definition 4.1. *A map $\mathcal{M} = \mathcal{S} \cup \mathcal{U} \cup \mathcal{R}$ is the union of the (claimed) safe region $\mathcal{S}$, the unknown region $\mathcal{U}$, and the (claimed) obstacle region $\mathcal{R}$. At the k-th timestep, the map $\mathcal{M}_k$ is **correct** if for all $p|^{B_k} \in \mathbb{R}^3$,*

$$p|^{B_k} \in \mathcal{S}_k|^{B_k} \implies p|^{B_k} \in \mathcal{F}|^{B_k}. \tag{4.8}$$

In words, $\mathcal{M}_k$ is *correct* if $\mathcal{S}_k$ is a subset of the free space $\mathcal{F}$ *when represented in the k-th body-fixed frame.*

The definition above is intentionally explicit about which reference frame various points and sets are represented in since this is the source of the main problem tackled in this section. Due to the odometry drift, there are two types of error common in state-of-the-art mapping algorithms:

(A) Errors in constructing the map: In current state-of-the-art implementations, the map is often represented computationally in the mapping frame M. Suppose at some time the robot detects an obstacle (relative to its body-fixed camera) at a position $p|^{B_k}$. It will update the map to mark this point as an obstacle:

$$\mathcal{S}_{k+1}|^M = \mathcal{S}_k|^M \backslash \{\widehat{T}^M_{B_k} \cdot p|^{B_k}\}, \tag{4.9}$$

that is, the estimated location $\widehat{T}^M_{B_k} \cdot p|^{B_k}$ is removed from the claimed free space. However,

notice that since the estimated transform $\widehat{T}^M_{B_k}$ is used instead of the true transform $T^M_{B_k}$, the location marked as an obstacle can be wrong. This problem is exacerbated since usually the line connecting the camera and $p|^{B_k}$ is also marked free, and therefore the wrong locations are marked as part of $\mathcal{S}_{k+1}$.

(B) Errors in querying the map: Now suppose the robot wants to navigate the environment. It must therefore check whether a point relative to the body-fixed frame $p|^{B_k}$ is free. To the best of our knowledge, all implementations will then check whether the corresponding point in the map, $\hat{p}|^M$, is a free point, that is they check whether

$$\hat{p}|^M = \widehat{T}^M_{B_k} \cdot p|^{B_k} \in \mathcal{S}_k|^M. \tag{4.10}$$

However notice again, since the estimated transform is used, this can lead to inconsistencies. In particular, owing to the odometry drift, the inconsistency will be worse when the obstacle point was inserted into the map many frames ago.[2]

In this section we overcome both such issues, *by ensuring the map is always correct in the body-fixed frame.* An equivalent perspective is that we update the maps such that despite using the estimated transform $\widehat{T}^M_{B_k}$ the map will be constructed and queried correctly.

The problem statement therefore is as follows:

Problem 4.1. *Consider a robotic system equipped with an RGBD camera operating in a static environment with obstacles $\mathcal{O} \subset \mathbb{R}^3$. Suppose an odometry module provides at each frame k the estimated odometry $\widehat{T}^{B_0}_{B_k} \in \mathbb{SE}(3)$, the relative odometry $\widehat{T}^{B_k}_{B_{k+1}} \in \mathbb{SE}(3)$ and a covariance of the relative odometry $\Sigma^{B_k}_{B_{k+1}} \in \mathbb{S}^6_{++}$. Suppose a mapping module can construct the best estimate map of the free space in the environment. Design a framework to correct the best-estimate map such that at each timestep, the map $\mathcal{M}_k$ is correct according to Definition 4.1* despite the odometry drift.

Note, we also assume that if a point $p \in \mathbb{R}^3$ is occupied, and within the camera's FoV, it will be detected as an obstacle. This is a common implicit assumption in the mapping literature. Note, an infrared depth camera often fails to detect transparent obstacles (e.g., windows and glass doors), or can fail if an obstacle has no texture that can be used by the stereo block-matching algorithms. Such issues are beyond the scope of this section.

In the next two sections, we will demonstrate how to construct correct maps by modifying existing baseline mapping algorithms. In particular we will extend (A) a mapping

[2]It will also becomes clear that time is not the only factor - points inserted/queried further from the robot will also be more inaccurate due to the larger moment arm that amplifies rotation errors. This is also why common heuristic algorithms of time- or distance-based forgetting cannot guarantee the correctness of the map. The methods proposed in this section will directly address such issues.

algorithm [107] which uses polytopes to represent the map of free space, and (B) the mapping algorithm [118] which uses signed distance fields to represent the free space. Both algorithms are depicted in Figure 4.2.

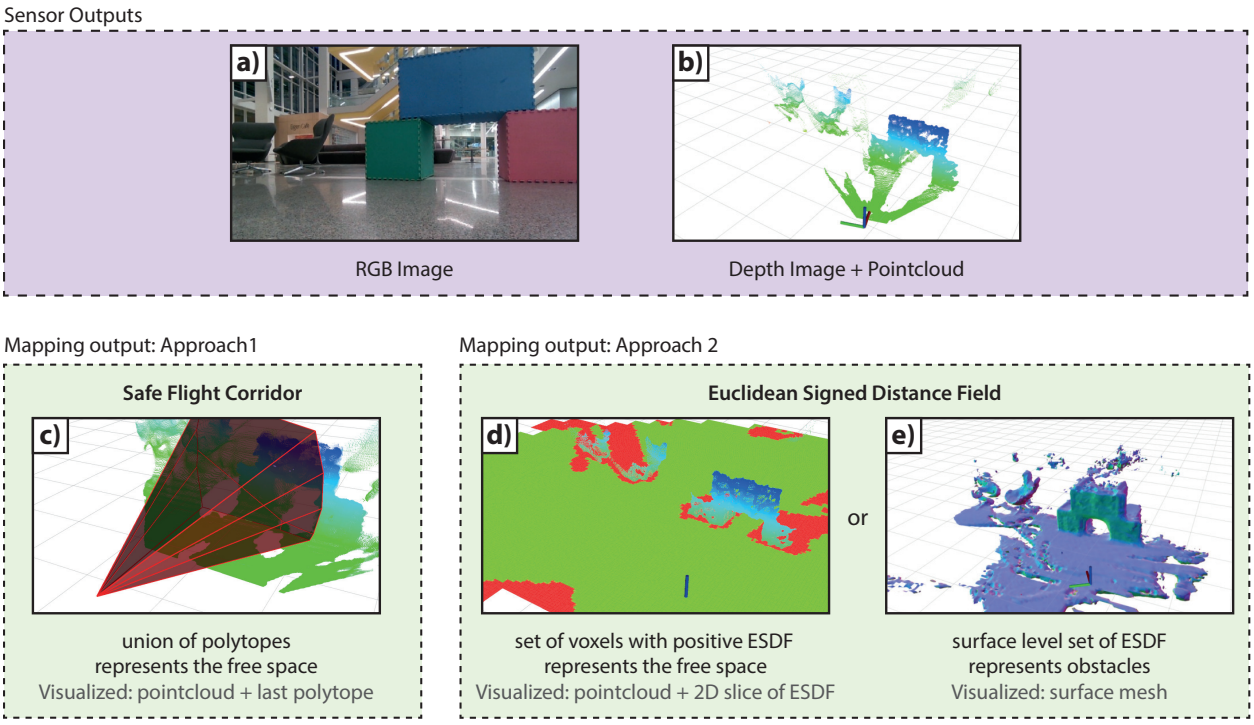

Figure 4.2: Two approaches to constructing an obstacle map. (Top row) An RGBD camera provides (a) the first person RGB image, and (b) the depth image/pointcloud constructed from stereo images. (Bottom row) The SFC approach represents the free space as a union of polytopes, one of which is depicted in (c). The ESDF approach represents the world using voxels, where each voxel stores the signed distance to the nearest obstacle. From this, both the (d) ESDF at specific voxels or (e) obstacle surface locations can be extracted and used for safe navigation. To aid the reader, in (c) and (d) the raw pointcloud is also visualized, and in (d) the color-scheme is such that voxels are marked green if $d > 0$, and red otherwise. This makes the map look binary, although it contains continuous values. Furthermore, note both methods operate in 3D - the 2D slice is used for visualization.

4.1.2 Approach 1: Certified Safe Flight Corridors

4.1.2.1 Background

In the first approach, one represents the obstacle-free region $\mathcal{S}_k$ at frame k as the union of n polytopes[3] (i.e., bounded and closed polyhedra),

$$\mathcal{S}_k|^{B_k} = \bigcup_{l=1}^{n} \mathcal{P}_k^l \tag{4.11}$$

where each polytope is of the form

$$\mathcal{P}_k^l = \{p \in \mathbb{R}^3 : A_k^l p \leq b_k^l\}, \tag{4.12}$$

This is often called the H-representation, since the polytope is defined by a set of half-space constraints [101]. An example of a polytope extracted from a depth image is shown in Figure 4.2c.

As the robot transitions from frame B_k to frame B_{k+1}, we can map each polytope from the previous frame to the new frame, and maintain the polytopes in the robot's body frame.

In the absence of odometry drift, one can directly compute the new polytopes:

$$\mathcal{P}_{k+1}^l = \{p \in \mathbb{R}^3 : A_{k+1}^l p \leq b_{k+1}^l\}, \tag{4.13a}$$
$$A_{k+1}^l = A_k^l R^T, \tag{4.13b}$$
$$b_{k+1}^l = b_k^l + A_k^l R^T t, \tag{4.13c}$$

using the estimated transforms

$$\widehat{T}_{B_k}^{B_{k+1}} = \begin{bmatrix} R & t \\ 0 & 1 \end{bmatrix}. \tag{4.14}$$

In the presence of odometry drift, however, the estimated transform $\widehat{T}_{B_k}^{B_{k+1}}$ is inexact, and this method fails to guarantee $\mathcal{P}_{k+1}^l \in \mathcal{F}$. Therefore, $\mathcal{M}_k$ is not guaranteed to be correct.

4.1.2.2 Proposed Approach

In the presence of odometry drift, since the transform $T_{B_k}^{B_{k+1}}$ is uncertain, the method in (4.13) does not work. Extending this approach to uncertain transforms is also not straightforward, since in the H-representation, an uncertain perturbation to a half-space does not result in

[3] n can be different at each k.

a new half-space. Here, we propose a novel method that uses the V-representation of the polytope to circumvent this issue.

In the V-representation, the polytope is the convex-hull of a set of vertices. Denote the set of vertices by

$$\mathcal{V}_i = \{v_{i,j}\}_{j=1}^{m_i} \subset \mathbb{R}^3, \tag{4.15}$$

where $v_{i,j} \in \mathbb{R}^3$ is the j-th vertex on the i-th face of a polytope.

We will use the V-representation to compute a new (deflated) polytope $\mathcal{P}_{k+1}$ from $\mathcal{P}_k$. The algorithm is described by the next Lemma and Theorem.

Lemma 4.1. *Suppose* $T_{B_k}^{B_{k+1}} \sim \mathcal{N}(\widehat{T}_{B_k}^{B_{k+1}}, \Sigma_k)$, *where*

$$\widehat{T}_{B_k}^{B_{k+1}} = \begin{bmatrix} R & t \\ 0 & 1 \end{bmatrix}. \tag{4.16}$$

Consider a polytope $\mathcal{P}_k$ *that is obstacle free,*

$$\mathcal{P}_k = \{p \in \mathbb{R}^3 : A_k p \leq b_k\} \tag{4.17}$$

where $A_k \in \mathbb{R}^{N\times 3}$, $b_k \in \mathbb{R}^N$. *Denote the* i*-th row as* $a_{k,i} \in \mathbb{R}^3$. *For each vertex* $v_{i,j} \in \mathcal{V}_i(\mathcal{P}_k)$ *on the* i*-th face of the polytope, define*

$$J_{i,j} = \begin{bmatrix} R & -R[v_{i,j}]_\times \end{bmatrix}, \quad \Sigma_{i,j} = \kappa J_{i,j} \Sigma_k J_{i,j}^T, \tag{4.18}$$

as in Assumption 4.1. Let each element of $\rho \in \mathbb{R}^N$ *be*

$$\rho_i = \max_{j\in\{1,\ldots,m_i\}} \sqrt{a_{k,i}^T \Sigma_{i,j} a_{k,i}} \tag{4.19}$$

Define a new polytope as

$$\mathcal{P}_{k+1} = \{p \in \mathbb{R}^3 : A_{k+1} p \leq b_{k+1}\}, \tag{4.20a}$$
$$A_{k+1} = A_k R^T, \tag{4.20b}$$
$$b_{k+1} = b_k + A_k R^T t - \rho. \tag{4.20c}$$

Given Assumption 4.1, $\mathcal{P}_k \in \mathcal{F}|^{B_k} \implies \mathcal{P}_{k+1} \in \mathcal{F}|^{B_{k+1}}$, *i.e., if* $\mathcal{P}_k$ *is obstacle-free, so is* $\mathcal{P}_{k+1}$.

Proof Sketch. [See Section 4.1.8.2 for the full proof.] It suffices to show that any obstacle

potentially on the boundary of $\mathcal{P}_k$ will not be in $\mathcal{P}_{k+1}$ after the rigid transform. To do so, we consider a potential obstacle on the i-th face of the polytope, and compute the ellipsoid the obstacle could be in after the transform. We compute the tangent plane of the ellipsoid normal to the i-th hyperplane, and compute the minimum shift necessary such that the shifted hyperplane does not contain the ellipsoid. We use the convexity of the polytope to show that the necessary shift on the i-th hyperplane is ρ_i, the maximum of the shifts necessary at each of the vertices on the i-th hyperplane of the polytope. This deflaion, when applied to each hyerplane of the polytope, guarantees that $\mathcal{P}_{k+1}$ does not contain the obstacle points. □

Finally, we can construct the main theorem.

Theorem 4.2. *Suppose the transform between frame is $T_{B_k}^{B_{k+1}} \sim \mathcal{N}(\widehat{T}_{B_k}^{B_{k+1}}, \Sigma_k)$. Given the k-th map is defined as in (4.11), define the $(k+1)$-th map as*

$$\mathcal{S}_{k+1}|^{B_{k+1}} = \bigcup_{l=1}^{N} \mathcal{P}_{k+1}^{l} \tag{4.21}$$

where each polytope is defined using Theorem 4.1. Then, given Assumption 4.1,

$$\mathcal{S}_k \subset \mathcal{F} \implies \mathcal{S}_{k+1} \subset \mathcal{F}, \tag{4.22}$$

that is, if $\mathcal{M}_k$ is correct by Definition 4.1, the updated map $\mathcal{M}_{k+1}$ will also be correct.

Proof. Directly apply Theorem 4.1 to each polytope in $\mathcal{S}_k$. □

In words, the theorem shows that when a each polytope in the map $\mathcal{M}_k$ is shrunk using Theorem 4.1, the new safe region $\mathcal{S}_{k+1}$ also remains certifiably obstacle-free. Once a given polytope has shrunk to zero volume, it can be forgotten entirely. Recall that as new camera frames are received, new polytopes can be constructed to define the free space in the operating environment and added to the set $\mathcal{S}_{k+1}$. We empirically study how quickly an environment deflates in Table 4.3 and in Section 4.1.8.8. Naturally, if the odometry covariance is smaller, the deflation rate is smaller Section 4.1.8.7.

Remark 4.1. *Compare (4.13) with (4.20). The two are identical except for the $-\rho$ vector in (4.20c). Each element $\rho_i \geq 0$, and therefore, this represents a shrinking operation. The net effect is that we transform the polytope by the estimated transform, but then shrink the polytope based on the odometry error covariance. Notice that this shrinking operation is tight: since there could exist an obstacle on the face of the polytope (indeed this is how they are constructed), the shrinking factor is the smallest allowable factor, by construction.*

Remark 4.2. *In implementation, notice that one needs to compute $\mathcal{V}_i(\mathcal{P}_k)$, the set of vertices, and then update the polyhedron by* (4.20c)*. Although this operation scales exponentially with the number of faces [64], efficient implementations exist, especially for 3D polytopes [101]. Empirically, we observe each polytope has on the order of 10-20 faces when using [107], and can be handled in real-time.*

4.1.3 Approach 2: Certified ESDFs

4.1.3.1 Background

The Euclidean Signed Distance Field (ESDF) is defined as the function $d : \mathbb{R}^3 \to \mathbb{R}$,

$$d(p) = \min_{o \in \mathcal{O}} \|o - p\|, \tag{4.23}$$

the minimum distance between the point p and all of the obstacles $\mathcal{O}$. A 2D slice of the ESDF is depicted in Figure 4.2d.

To evaluate (4.23), o and p must be expressed in a common frame, commonly referred to as the mapping frame. Since this is done in the mapping frame, it is denoted as the function $d_M : \mathbb{R}^3 \to \mathbb{R}$. The claimed-safe region $\mathcal{S}_k$ is therefore

$$\mathcal{S}_k = \{p \in \mathbb{R}^3 : d_M(p) \geq 0\} \tag{4.24}$$

For safety-critical path planning and control, we need the ESDF at points relative to the body-fixed frame. The common approach is to assume the odometry estimate is exact, and determine $d(p|^{B_k})$ by expressing it in the map frame and evaluating d_M:

$$d(p|^{B_k}) \approx d_M(\widehat{T}^M_{B_k} \cdot p|^{B_k}) \tag{4.25}$$

However, since the estimate $\widehat{T}^M_{B_k}$ is inexact, this method can lead to over- or under-estimates. Overestimated distances are unsafe since they could lead to collisions.

4.1.3.2 Proposed Approach

The goal is to construct an ESDF that is safe, i.e., underestimates the distance to obstacles. Using Definition 4.1, a *Certified-ESDF* is defined as

Definition 4.2. *Let the obstacle set be $\mathcal{O} \subset \mathbb{R}^3$, assumed static in frame I. Let the ESDF of $\mathcal{O}$ be $d : \mathbb{R}^3 \to \mathbb{R}$. A* ***Certified-ESDF*** *(C-ESDF) at timestep k is a function $d^k_M : \mathbb{R}^3 \to \mathbb{R}$,*

such that for all points $p|^{B_k} \in \mathbb{R}^3$,

$$d(p|^{B_k}) \geq d_M^k(\widehat{T}_{B_k}^M \cdot p|^{B_k}) \tag{4.26}$$

where $\widehat{T}_{B_k}^M \in \mathbb{SE}(3)$ *is the estimated rototranslation between* B_k *and* M.

Comparing (4.25) with (4.26), the goal of certification is to change the $\approx$ into $\geq$. That is, a Certified-ESDF is one where for any body-fixed point $p|^{B_k}$, if the point is expressed in the mapping frame *using the estimated rototranslation*, we have *underestimated* the distance to the nearest obstacle:

$$\underbrace{d(p|^{B_k}) = \min_{o \in \mathcal{O}} \left\| p|^{B_k} - o|^{B_k} \right\|}_{\text{true ESDF}} \geq \underbrace{d_M(\widehat{T}_{B_k}^M \cdot p|^{B_k})}_{\text{estimated ESDF}}. \tag{4.27}$$

To accomplish this, we propose a strategy of deflating the ESDF. We derive a recursive guarantee to ensure the ESDF remains certified for all k.

Theorem 4.3. *Suppose at timestep* $k \in \mathbb{N}$, *the ESDF* $d_M^k : \mathbb{R}^3 \to \mathbb{R}$ *is a Certified-ESDF. Let the rototranslation between frames be* $T_{B_{k+1}}^{B_k} \sim \mathcal{N}(\widehat{T}_{B_{k+1}}^{B_k}, \Sigma_k)$. *Let the* $d_M^{k+1} : \mathbb{R}^3 \to \mathbb{R}$ *be defined by*

$$d_M^{k+1}(p|^M) = d_M^k(p|^M) - \sqrt{\lambda_{\max}(\Sigma_p)} \tag{4.28}$$

for all $p|^M \in \mathbb{R}^3$, *where*

$$\widehat{T}_{B_{k+1}}^{B_k} = \begin{bmatrix} R & t \\ 0 & 1 \end{bmatrix}, \tag{4.29a}$$

$$J = \begin{bmatrix} R & -R[\widehat{T}_M^{B_{k+1}} \cdot p|^M]_\times \end{bmatrix}, \tag{4.29b}$$

$$\Sigma_p = \kappa J \Sigma_k J^T. \tag{4.29c}$$

and $\kappa > 0$ *is as defined in Assumption 4.1. Given Assumption 4.1,* d_M^{k+1} *is also a Certified-ESDF at timestep* $k+1$.

Proof Sketch. [See Section 4.1.8.3 for the full proof.] Consider any point $p|^{B_{k+1}}$ and evaluate the potential positions it could correspond to in frame B_k. This is an ellipsoid as in Assumption 4.1, and therefore the ESDF at $p|^{B_{k+1}}$ must be the minimum of all of the ESDF values for the corresponding points in the ellipsoid. Since, by definition, the Lipschitz constant of an ESDF is one, this minimum ESDF can be lower bounded by the ESDF at the center

minus the radius of the smallest sphere containing the ellipsoid. We use the eigenvalues of the ellipsoid to compute the radius of sphere, arriving at the expression. □

Remark 4.3. *Notice that the correction is* $-\sqrt{\lambda_{max}(\Sigma_p)}$ *in* (4.28) *(different for each p). As with the certified SFCs, this is a deflation operation that decreases the estimated distance to an obstacle.*

Remark 4.4. *The implementation of this deflation operation is remarkably simple and easily parallelized on a GPU. In our implementations, we added an additional deflation integrator to the code in [118]. At each frame, when the relative odometry with covariance is received, we can compute the deflation at each voxel in parallel using* (4.28).

4.1.4 Safe Navigation with Certified Maps

Here we summarize the key ideas presented in this section, and suggest strategies to leverage certified maps to achieve safe navigation.

A fundamental principle of our approach is ensuring that maps remain correct with respect to the body-fixed frame. To achieve this, we deflate the safe regions of the map based on the incremental odometry error at each timestep. The required deflation has an analytic expression.

Our implementation is as follows. When the $(k+1)$-th camera frame is received from the sensor, we compute the odometry estimate, and its relative covariance. Next, we apply the deflation step using the proposed algorithms. Finally, we incorporate new safe regions identified by the depth image to assimilate new information while discarding regions that can no longer be certifiably correct.

One can also maintain both the baseline and certified maps in memory simultaneously. While the memory usage increases, the certified maps tend to be smaller than the full map, maintaining both maps offers significant advantages. In particular, our certified mapping methods can integrate naturally with existing safety filtering methods like [10, 164]. These methods generate nominal trajectories to achieve mission objectives, but use a backup trajectory to ensure that the robot can safely stop based on the currently available information. In our framework, one can use the baseline map for nominal trajectory planning, but use the certified map for collision and safety checks. This combination enables agile motion while strictly guaranteeing safety.

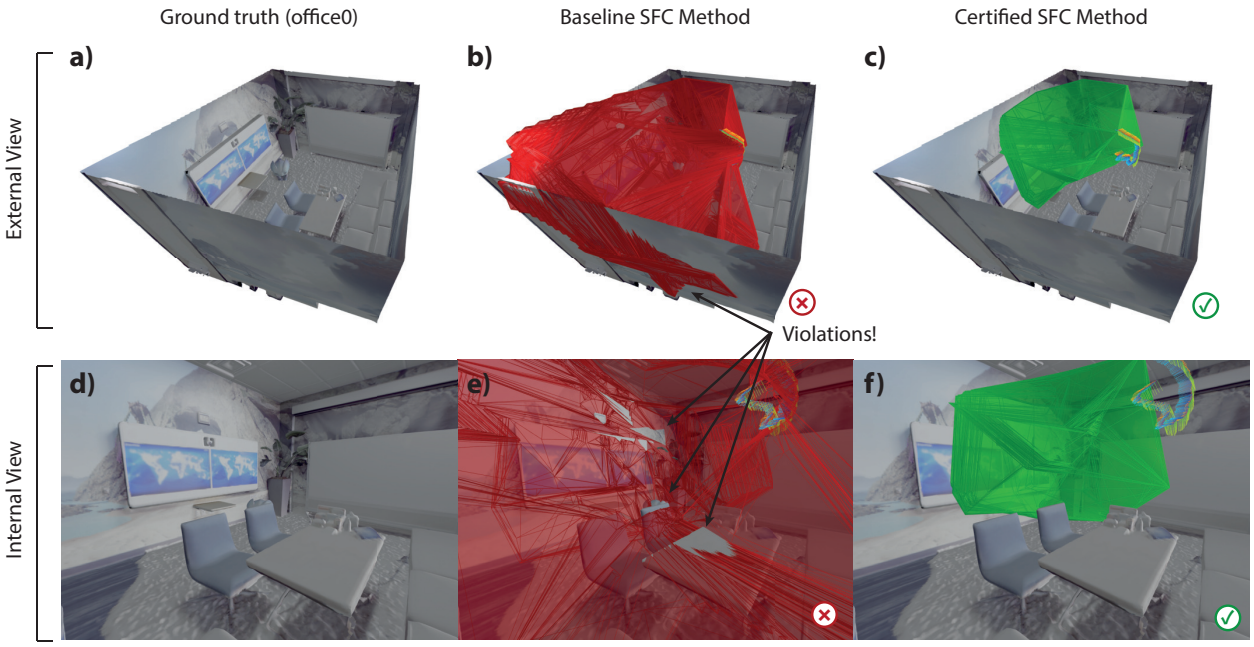

Figure 4.3: Visualization of a snapshot of the `office0` environment mapped using the baseline and certified SFC methods. (a, d) shows the `office0` environment, while (b, e) and (c, f) show the respective $\mathcal{S}$ sets at the 500-th timestep from an external and an internal view. The baseline map claims a larger volume to be safe compared to the certified method (red volume is larger than green volume). However, we can also see numerous regions where the red region intersects with the ground truth mesh, indicating that the claimed safe region contains obstacle points. In the certified method, we see no violations.

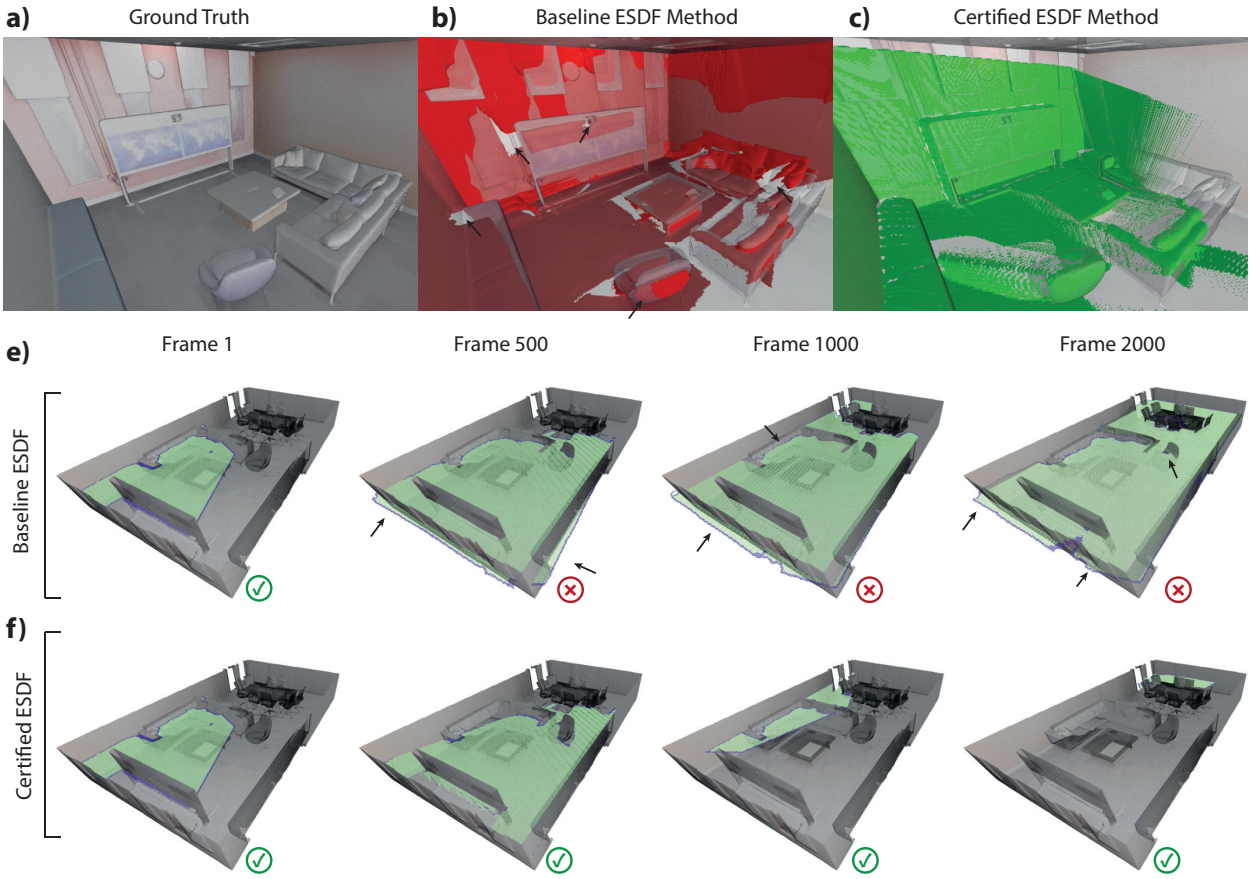

Figure 4.4: Visualization of the maps generated using the baseline and certified ESDF methods on the `office3` environment. In (a) we see the ground-truth mesh. In (b) and (c) we can see the internal view after 500 timesteps. As in Figure 4.3, although the baseline method maps a larger volume (red mesh is larger than green mesh), it also contains many violations. In (e) and (f) we see a slice of the ESDF over time. The green region indicates the $\mathcal{S}$ set at the respective times. The small black arrows point to various violations in the baseline method, while in the certified methods we see no violations.

4.1.5 Simulations

We present results on the accuracy and correctness of both approaches for certified mapping presented above. As a reminder, the goal is to demonstrate that despite odometry drift, the region reported by our algorithms to be a part of the free space is indeed obstacle-free. First, we evaluate the performance of both the Certified SFCs and the Certified ESDFs methods on the Replica dataset (described below) and compare it to various baselines. Second, we have run hardware experiments with a rover, and show that by considering the certification bound the rover can avoid collisions. Additional results are reported in Section 4.1.8.6 and Section 4.1.8.7.

Evaluation Method

We evaluated the performance of our implementations on the Replica dataset [158], with ground-truth trajectories generated as in [188]. From the ground-truth trajectory the RGBD image sequence was generated, and used as the inputs to the mapping algorithms. We perturbed the trajectory to generate the estimated trajectory from a simulated odometry system as follows:

$$\widehat{T}^{B_k}_{B_{k+1}} = T^{B_k}_{B_{k+1}} \operatorname{Exp}(\tau), \quad \tau \sim \mathcal{N}(0, \Sigma) \tag{4.30}$$

where $T^{B_k}_{B_{k+1}} \in \mathbb{SE}(3)$ is the transform between subsequent frames of the ground-truth trajectory of the camera and $\widehat{T}^{B_k}_{B_{k+1}} \in \mathbb{SE}(3)$ is the estimated transform between subsequent frames used in the mapping algorithms. For our experiments we used $\Sigma \in \{1e-5I, 1e-6I\}$. Evaluating the Absolute Translation Error (ATE) as in [184], the generated trajectories had between $1-3\%$ ATE, inline with the performance of state-of-the-art Visual Inertial Odometry (VIO) methods. Each trajectory was 2000 frames long, running at 30 FPS.

Baselines

We compared our proposed certified approaches to the following mapping methodologies:

(A) *Baseline SFC* - At each camera frame, the depth map is used to construct a point-cloud of obstacles within the current field of view. From this a convex polyhedron is extracted, and appended to a list of safe polyhedrons. The union of these polyhedrons is considered the safe flight region. We used the library [107] to perform the convex decomposition.

(B) *Heuristic SFC* - This is the same algorithm as in (A), except that a time-based forgetting mechanism is introduced, as is common in robotic mapping implementations. In

particular, we only keep the last 60 frames (2 seconds) of polyhedrons when constructing the safe flight region.

(C) *Baseline ESDF* - At each camera frame, the depth map is used to update the Truncated Signed Distance Field (TSDF) of the environment. At regular intervals a wave propagation algorithm constructs/updates the ESDF of the environment. Regions with positive ESDF are considered part of the safe flight region. We used the library [118] to construct the TSDF and ESDF.

(D) *Heuristic ESDF* - This is the same algorithm as in (C), except that a distance-based forgetting mechanism is introduced. In particular, we forget any TSDF and ESDF voxels that are more than 3 m away from the camera.

These baselines will be compared to the certified methods proposed in this section:

(E) *Certified SFC* - This is the same algorithm as in (A), except that at each frame, each polytope is deflated as described in Section 4.1.2.

(D) *Certified ESDF* - This is the same algorithm as in (C), except that at each frame, the ESDF is deflated as described in Section 4.1.3.

Metrics

To evaluate the performance of the methods, we consider three metrics:

(I) *Violation Rate:* The violation rate measures the percentage of ground-truth mesh points that (incorrectly) lie within the claimed free space. The violation rate should be close to 0%.

(II) *Maximum Violation Distance:* For any violating point we measure the maximum distance of the violation, i.e., how far into the claimed free space is an obstacle point. The violation distance should be close to 0 mm. If there are no violating points, the violating distance is 0 mm.

(III) *Free-Space Volume:* This measures the total volume of the space that is claimed to be free. The free-space volume should be as large as possible.

Results

Tables 4.1, 4.2, and 4.3 summarize the results from the simulations. Figure 4.3 and Figure 4.4 visualize the results and qualitatively show the behavior of the proposed methods.

Table 4.1: Violation Rates. This table summarizes the fraction of violating ground-truth obstacle points for each environment and algorithm. This table shows results with Σ =1e-6I.

Algorithm	Violation Rates (%)							
	office0	office1	office2	office3	office4	room0	room1	room2
Baseline SFC	18.60 %	12.76 %	10.13 %	12.74 %	14.44 %	10.74 %	19.17 %	6.85 %
Heuristic SFC	0.11 %	0.57 %	0.09 %	0.10 %	0.27 %	0.02 %	0.39 %	0.92 %
Certified SFC	0.0002%	0.0047%	0.0008%	0.0005%	0.0014%	0.0002%	0.0009%	0.0012%
Baseline ESDF	48.15 %	35.31 %	51.51 %	54.66 %	48.35 %	62.03 %	48.15 %	47.49 %
Heuristic ESDF	31.55 %	34.39 %	7.63 %	4.66 %	10.08 %	9.25 %	20.88 %	16.32 %
Certified ESDF	0.5443%	0.0610%	0.0809%	0.0227%	0.0538%	2.4259%	0.0149%	0.0519%

Table 4.2: Maximum Violation Distance. This table summarizes the distance by which violating ground-truth obstacle points penetrate the estimated free space for each environment and algorithm. This table shows results with Σ =1e-6I.

Algorithm	Maximum Violation Distance (mm)							
	office0	office1	office2	office3	office4	room0	room1	room2
Baseline SFC	102.7	95.3	159.7	177.6	125.5	117.1	191.4	85.0
Heuristic SFC	22.1	14.5	18.4	11.6	8.9	11.0	14.2	12.8
Certified SFC	0.0	0.9	0.4	0.9	1.7	0.9	0.7	0.7
Baseline ESDF	604.3	406.9	520.0	671.1	636.9	990.8	604.6	594.0
Heuristic ESDF	563.6	379.5	311.8	429.4	366.6	428.5	384.7	435.4
Certified ESDF	109.5	82.5	141.4	100.0	66.3	120.0	100.0	82.5

Table 4.3: Estimated Free Space Volume. This table summarizes the volume of the estimated free space at the end of the simulation for each environment and algorithm. This table shows results with Σ =1e-6I.

Algorithm	Estimated Free Space Volume (m^3)							
	office0	office1	office2	office3	office4	room0	room1	room2
Baseline SFC	34.8	17.6	40.8	56.6	63.3	53.0	38.7	29.4
Heuristic SFC	6.7	3.6	4.3	4.6	15.7	12.3	6.9	7.5
Certified SFC	5.7	2.6	3.6	3.0	12.5	9.1	5.8	4.4
Baseline ESDF	46.1	23.2	77.5	110.9	99.7	105.4	53.8	63.6
Heuristic ESDF	39.5	23.0	31.3	42.0	51.5	28.6	34.5	38.7
Certified ESDF	10.7	3.8	6.2	5.0	14.3	31.5	6.6	4.5

Figure 4.3 visualizes one of the runs from the `office0` environment. Figures (a, d) shows the ground-truth mesh of the environment from two different views. In (b, e) we see the safe flight polytopes in the baseline method visualized as the red region. One can see that the red region clearly intersects with the ground-truth mesh, and each intersection represents a violation. The violations are particularly noticeable for regions that were mapped further in the past, and from non-convex and thin obstacles like the chair or table surfaces. In contrast, in (c, f) we see the safe flight polytope from the proposed certified algorithms, drawn as the green region. We can see that the green region is smaller than the red polytope, but it also contains no violating points (see also Table 4.2 and Table 4.3). Effectively, we can see that due to the odometry drift, the algorithm cannot be confident about the exact location of, for example, the chair and the desk, and therefore these regions were removed from the map. Although the volume of free space is smaller, the map is guaranteed to be correct.

From Table 4.1 we can observe that both certification methods significantly reduce the number of violations. In the baseline methods, the violation rates are between 6 and 60%, while in the certified methods, the violation rates are between 0-3%. Note, we cannot expect the certified methods to have exactly zero violations, since we are using the truncated noise model for odometry. Nonetheless, empirical performance of the certified methods still shows that the proposed methods can effectively avoid classifying obstacle regions as free.

Furthermore, we can see that although the heuristic forgetting methods can also reduce the number of violations, the level of reduction is hard to control. Since the forgetting factor is tuned heuristically and independently of the true noise level in the system, it can sometimes lead to good rejection of obstacles (as in the SFC method) or poor rejection of obstacles (as in the ESDF).

From Table 4.2 we observe that the maximum distance a violating point intersects the map is also reduced using the certified methods. We see that the maximum violation is sub-millimeter for the SFC methods, demonstrating a reduction of 2 orders of magnitude compared to the baseline. In the ESDF approaches, we still see a significant reduction in the maximum violation distance (about an order of magnitude reduction), although there are some violations on the order of 100 mm. This seems to be a limitation of the ESDF approach, since the ESDFs are represented using discrete voxels computationally. We chose a voxel size of 20 mm, and therefore the violations are on the order of 1-5 voxels of error.[4]

The source of this larger error is likely the dataset itself. We have checked which voxels are causing these large errors, and it seems to be the voxels that are close to non-manifold

[4]Finer grid resolution can help, but will increase the computational and memory requirements. As a sense of scale, each environment is on the order of $6 \times 6 \times 3$ m, and therefore has approximately $300 \times 300 \times 150$ voxels. See Table 4.4 for additional details.

surfaces in the Replica dataset, for instance near the leaves of plants, or around table/chair legs, which are thin and long. Near these surfaces, the raw data is inconsistent, and we suspect that it leads to higher error rates than expected.

Finally, if we consider the volume of free space mapped in Table 4.3, we can see that due to the certification, the volume of the estimated free space is lower for the certified methods than it is for the heuristic or baseline methods. However, since the violation rate of the uncertified methods is significant, the free space cannot be trusted for path planning around obstacles. Despite the smaller volume of free space, the certified methods allow the full region to be trusted when used in planning (Section 4.1.8.8).

Comparing the SFC and ESDF methods, in the results presented the SFC methods seem superior, since they have fewer violations, and the violating points violate the free space by a smaller distance. However this does come at the expense of expressiveness and computational cost. The SFC methods require the use of unions of convex polytopes to represent the free space, and in cluttered environments can sometimes lead to very small volumes of free space. The ESDF implementations are also more mature, with implementations like [118] allowing for efficient use of a GPU, which allows the ESDF to be computed more efficiently than the SFC.

4.1.6 Rover Experiments

In this section we demonstrate the utility of the proposed certified mapping frameworks in ensuring a robot can safely navigate an environment. We demonstrate that when a rover is tasked to navigate through an environment, and in particular reverse blindly into a region it previously mapped, the accumulated odometry error can lead to the rover colliding with previous mapped obstacles. Instead, by using the proposed methods, the rover will avoid traversing into regions that it can no longer certify are obstacle-free. Additional experiments are reported in Section 4.1.8.8.

Experimental Setup

A block diagram of the experimental setup is shown in Figure 4.5a). We use a ground rover, the AION R1 UGV equipped with an Intel Realsense D455 camera. All perception, planning, and control is executed on the onboard computer, an Nvidia Orin NX 16GB. The Realsense camera sends stereo infrared images to the Orin NX at 30FPS. A state-of-the-art visual slam algorithm (Nvidia IsaacROS Visual SLAM) is used to compute the odometry estimate. The Realsense camera also produces a depth image, which is sent to the obstacle mapping library (an adapted version of Nvidia IsaacRos NvBlox) which constructs an ESDF

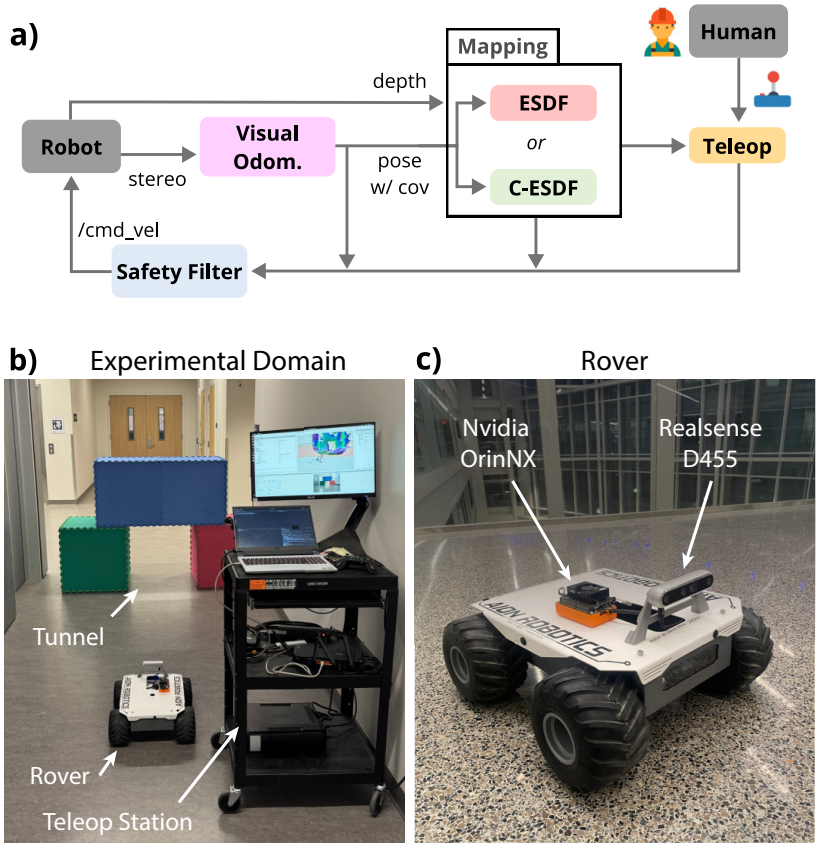

Figure 4.5: Rover Experimental Setup. (a) Block diagram. The human is teleoperating the rover using only the FPV feed and the reconstructed obstacle map computed and streamed in real-time. The map is also used onboard the robot to stop the robot if it violates safety constraints. The safety filter can either use the baseline ESDF or the Certified ESDF. (b) Picture of the testing environment. The robot drives through the tunnel, mapping it as it passes through. After exploring the corridors, the rover tries to return through the tunnel in reverse, without remapping the tunnel. (c) shows the rover in more detail. The AION R1 UGV has been modified, with all sensing on Intel Realsense D455, and all compute on the Nvidia OrinNX 16GB.

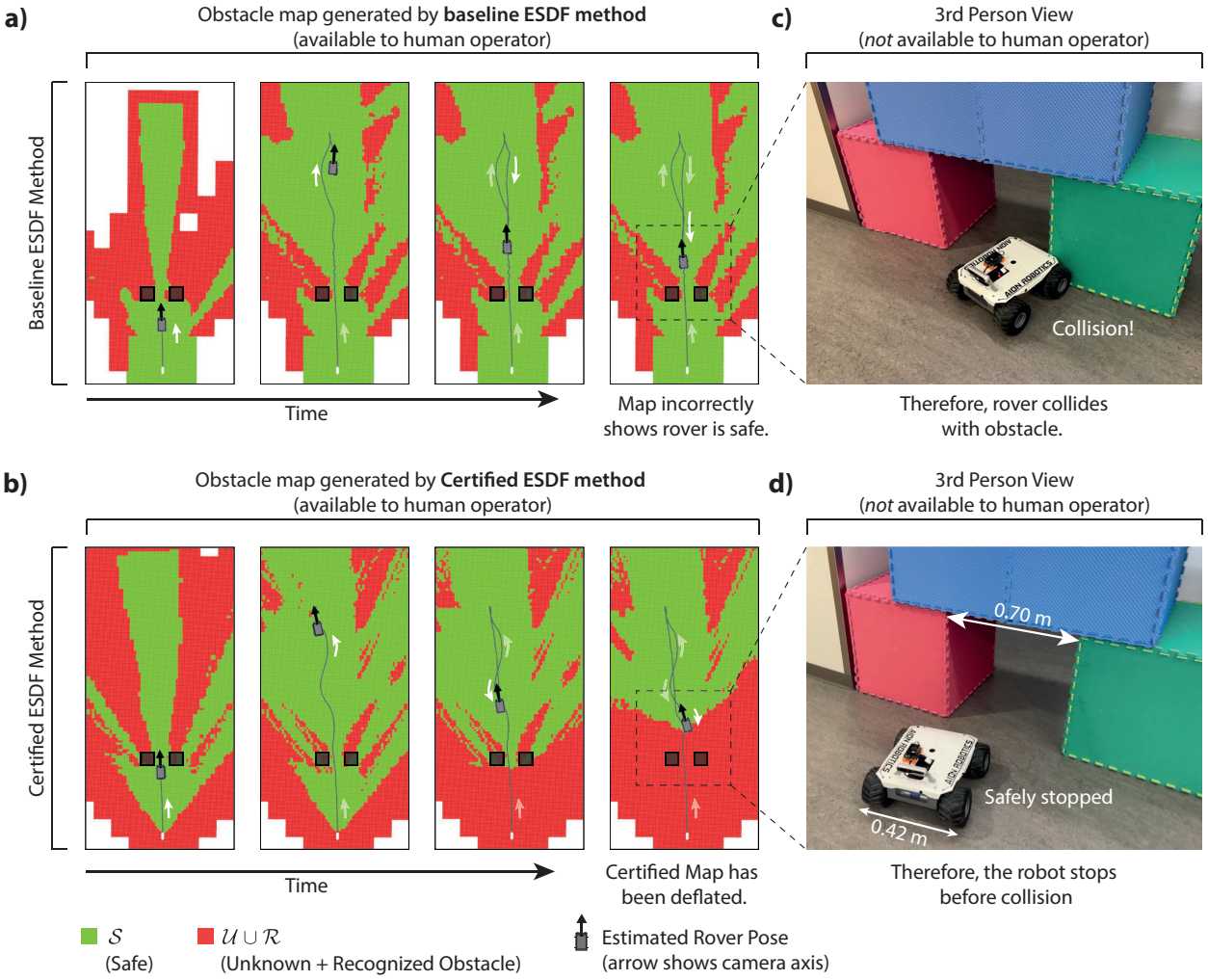

Figure 4.6: Rover Experimental Results. (a, b) shows snapshots of the reconstructed obstacle map and the estimated rover pose with the baseline method (a) and the certified method (b). This is the view presented to the human teleoperating the robot. Note, two small black boxes are drawn in each frame (in post) to indicate to the reader the location of the red and green boxes during the experiment. These were not visible to the human operator during the experiments. (c, d) show the final state of the robots at the end of the trajectory. In (c), the baseline method the robot has crashed with the green obstacle, although looking at the last panel of (a), we can see that the robot thinks it is in the middle of the tunnel in the free space. In (d), we see the robot stopped 15 cm before crashing with the red obstacle, and this is because the map has been deflated sufficiently that the safety filter prevents the robot from continuing backwards. Notice between the second, third and fourth frames in (b) the green regions near the bottom change into red regions, indicating the Certified ESDF cannot certify that the red region is obstacle-free.

of the environment in real-time. All parameters and code is available at [redacted].

A human operator uses a joystick to send desired linear and angular velocities to the robot. Using the constructed ESDF, a safety filter forward propagates the robot's state under a desired command a short (0.5 s) horizon into the future and checks whether the trajectory lies strictly within $\mathcal{S}_k$. If so, the command is sent to the robots' motor controllers. If not, the safety filter zeros the linear command, and sends a reduced angular speed command. This allows the robot to continue to spin to acquire new information about the environment, without physically moving and potentially colliding with the obstacles. The safety filter was tuned offline to ensure that in the absence of odometry drift, the robot stops within 15 cm of the obstacle both when driving forwards or backwards.

To compute the certified-correct map, we use the techniques of Section 4.1.3 to compute the certified ESDF representing the local geometry. To correctly deflate the ESDF, we require the odometry estimate, and the covariance of the incremental transform between successive camera frames, i.e., of $\widehat{T}^{B_k}_{B_{k-1}}$.

To the best of the author's knowledge however, this information is not reported by any state-of-the-art odometry/pose estimation algorithms. Most algorithms (including Nvidia's vSLAM) only report the covariance of the odometry estimate between the initial frame and the current frame, i.e., of $\widehat{T}^{B_k}_{B_0}$. In [111] the authors computed the covariance of relative poses after solving a pose-graph optimization problem by using the Jacobian of the local solution (see [111, Section IX.B] for details). However this only allows one to find the covariance of relative transforms between keyframes, and does not allow one to find the relative transform between successive camera frames.

Here, we use the following method to estimate the covariance between relative frames. VSLAM reports the odometry estimates $\widehat{T}^{B_k}_{B_0}$, $\widehat{T}^{B_{k+1}}_{B_0}$, and the associated covariances $\Sigma^{B_k}_{B_0}$, $\Sigma^{B_{k+1}}_{B_0}$. Assuming $T^{B_k}_{B_0}$ and $T^{B_{k+1}}_{B_0}$ are highly correlated since they are successive frames, we can define a correlation coefficient $\rho \in [-1, 1]$ (we use $\rho = 0.99$) between these camera frames. We can then estimate the covariance of the relative transform $\Sigma^{B_{k+1}}_{B_k}$ along the lines of [111]. The analysis is presented in Section 4.1.8.4.

Experimental Results

Figure 4.6 summarizes the results of the rover experiments, with additional trials available in the supplementary video, all demonstrating similar outcomes.

The human operator's task was to navigate the rover without line-of-sight through a narrow tunnel, explore and map the environment, and then return to the starting location by reversing through the tunnel. The rover was intentionally reversed through the tunnel to avoid re-mapping the obstacle geometry, forcing it to rely on its previously constructed

maps for navigation.

Snapshots in Figure 4.6a show the baseline mapping method. Initially, the tunnel and the surrounding corridors are mapped accurately. As the operator tries to reverse through the tunnel the final snapshot suggests that the rover is well aligned with the tunnel and positioned safely within the green region $\mathcal{S}$. However, despite this seemingly safe alignment, the rover collided with an obstacle Figure 4.6c, a failure in the baseline mapping approach.

In contrast, our proposed method deflates the safe regions in response to the odometry drift. In Figure 4.6b, the map initially classifies a large region as safe (green). However, as rover reverses to the tunnel, the deflation has caused parts of the map to turn red, indicating that these areas can no longer be certified to be obstacle free. Indeed, when the rover reaches the boundary between red and green regions, the safety filter prevents further motion, successfully preventing collision.

The same behavior was consistently observed across multiple trials with different trajectories. Additional discussion on the effect and extent of the deflation is presented in Section 4.1.8.8.

4.1.7 Conclusions

Limitations and Future Directions

While the proposed methods are provably correct, they rely on key assumptions, particularly Assumption 4.1, which truncates the normal distribution of pose perturbations to bound the effects of a rototranslation on an obstacle point. Although this simplification facilitates our framework, it may not hold in practice. Methods such as those in [26, 111] could improve these approximations and warrant further exploration.

Additionally, we assumed that incremental odometry perturbations follow a normal distribution in the Lie algebra of $\mathbb{SE}(3)$. However, this assumption may not hold in practice, especially in the presence of outliers (see e.g. [179]). A valuable direction for future work is to rigorously characterize the error distribution of odometry systems, both analytically and empirically.

We also highlight the need for perception algorithms to estimate and report the uncertainty of incremental pose transforms, rather than overall pose error/covariance, which grow unbounded without successful loop closures. Metrics such as relative translation and rotation errors [184] or the correlation between pose uncertainties (as in [111]) should be computed and reported. In lieu of this, our experiments estimated incremental pose error covariances using the method described in Section 4.1.8.4. For certifiability guarantees, going forward we will need odometry algorithms capable of directly reporting the incremental pose error

covariance.

Our algorithm intentionally deflates the map, and this reduces the navigable volume for the robot. It is challenging to estimate how much the volume reduces prior to a mission, since the deflation depends on the exact obstacle geometry, features used by the odometry algorithm, and the speed of the robot (which affects how quickly new parts of the environment are observed). Empirically, we have shown that as the odometry covariance decreases, the volume of the free space increases, and approaches the volume of baseline methods in the error-free case (Section 4.1.8.7). We also operated our rover in a larger room, and in Section 4.1.8.8 we show empirically that the certified methods can yield similar or larger volumes of free space than the heuristic method. Further analysis into this warranted.

Beyond odometry drift, there are other sources of error that can invalidate the correctness of the map - the operating environment and each subsystem can introduce errors that are hard to correct or even detect. For instance, depth estimation algorithms (e.g., block-matching methods) can fail under conditions like glass surfaces or featureless walls. Similarly, communication/computational latencies can introduce errors that are hard to characterize with the current framework.

Summary

As robots increasingly operate in unstructured environments, the importance of tightly integrated perception, planning, and control systems becomes evident. Our experiments demonstrate that even over short distances, perception inaccuracies due to odometry drift can lead to unsafe behaviors, including collisions.

This section presents a step toward building perception modules that not only generate accurate state estimates and obstacle maps but also provide correctness guarantees. Specifically, if the incremental odometry error per frame can be bounded, our framework modifies (or deflates) obstacle-free regions in a map such that it remains correct at all times with respect to the robot's body frame.

We proposed two methods for implementing these corrections based on different map representations: (I) Certified SFCs, and (II) Certified ESDFs. By constructively proving the correctness of these methods, we developed algorithms that guarantee safe map modifications. Extensive simulations using high-quality datasets, along with real-world experiments on a robotic rover, validate the effectiveness of our approach in creating certifiably-correct maps.

A key insight from our rover experiments is the demonstration of failure modes in state-of-the-art mapping methods. Unlike typical demonstrations, where robots map regions within the camera's field of view or use 360° sensors (e.g., LIDAR), we intentionally operated the

robot in its blind spot to highlight the challenges posed by accumulated odometry drift. Our proposed methods successfully mitigated these issues, preventing collisions and ensuring safe navigation.

4.1.8 Appendix

4.1.8.1 Review of Matrix Lie Groups

Here we review the fundamentals of representing a pose and its uncertainty through the language of Lie groups and Lie algebras. We refer to readers to [26, 111, 155] and references therein for a more complete description.

The Lie group $\mathbb{SO}(3)$ is the set of valid 3D rotation matrices, and the group $\mathbb{SE}(3)$ is the set of rigid transformations in 3D:

$$\mathbb{SO}(3) = \left\{ R \in \mathbb{R}^{3\times 3} : RR^T = I_3, |R| = 1 \right\},$$
$$\mathbb{SE}(3) = \left\{ T = \begin{bmatrix} R & t \\ 0 & 1 \end{bmatrix} \in \mathbb{R}^{4\times 4} : R \in \mathbb{SO}(3), t \in \mathbb{R}^3 \right\}.$$

Both $\mathbb{SO}(3)$ and $\mathbb{SE}(3)$ are matrix Lie groups, i.e., the group composition operation is the standard matrix multiplication operation.

The group action for $\mathbb{SE}(3)$ is $\cdot : \mathbb{SE}(3) \times \mathbb{R}^3 \to \mathbb{R}^3$, which transforms a point p from its representation in frame A to that in frame B. Given $T_A^B = \begin{bmatrix} R & t \\ 0 & 1 \end{bmatrix} \in \mathbb{SE}(3)$,

$$p|^B = T_A^B \cdot p|^A = Rp|^A + t. \tag{4.31}$$

The tangent space centered at identity is called the Lie algebra of a Lie group. The Lie algebra is a vector space of all possible directions an element of the group can be perturbed locally. The Lie algebras of $\mathbb{SO}(3)$ and $\mathbb{SE}(3)$ are denoted $\mathfrak{so}(3)$ and $\mathfrak{se}(3)$ respectively:

$$\mathfrak{so}(3) = \left\{ \omega \in \mathbb{R}^{3\times 3} : \omega^T = -\omega \right\},$$
$$\mathfrak{se}(3) = \left\{ \begin{bmatrix} \omega & \rho \\ 0 & 0 \end{bmatrix} \in \mathbb{R}^{4\times 4} : \omega \in \mathfrak{so}(3), \rho \in \mathbb{R}^3 \right\}.$$

These vector spaces are isomorphic to the Euclidean vector space $\mathbb{R}^3$ and $\mathbb{R}^6$ respectively. The $\wedge$ operator converts the Euclidean vector to an element of the Lie Algebra. For $\mathbb{SO}(3)$,

$\wedge : \mathbb{R}^3 \to \mathfrak{so}(3)$:

$$\phi^\wedge = \begin{bmatrix} \phi_1 \\ \phi_2 \\ \phi_3 \end{bmatrix}^\wedge = \begin{bmatrix} 0 & -\phi_3 & \phi_2 \\ \phi_3 & 0 & -\phi_1 \\ -\phi_2 & \phi_1 & 0 \end{bmatrix} \tag{4.32}$$

while for $\mathbb{SE}(3)$, $\wedge : \mathbb{R}^6 \to \mathfrak{se}(3)$:

$$\xi^\wedge = \begin{bmatrix} \rho \\ \phi \end{bmatrix}^\wedge = \begin{bmatrix} \phi^\wedge & \rho \\ 0 & 0 \end{bmatrix}. \tag{4.33}$$

The $\vee$ operator performs the inverse of $\wedge$.

Given an element of the Lie algebra, we can convert it to the corresponding element of the group using the exponential map. For $\mathbb{SE}(3)$, the exponential map is $\exp : \mathfrak{se}(3) \to \mathbb{SE}(3)$,

$$\exp(X) = \sum_{k=0}^{\infty} \frac{X^k}{k!} = I + X + \frac{X^2}{2} + \cdots \tag{4.34}$$

For convenience, we also define the Exp map, which maps from the Euclidean representation directly to the group element, $\mathrm{Exp} : \mathbb{R}^6 \to \mathbb{SE}(3)$,

$$\mathrm{Exp}(\xi) = \exp(\xi^\wedge). \tag{4.35}$$

Analytic expressions for this are provided in [155, Appendix]. The corresponding inverse operations are log and Log.

The adjoint matrix of $\mathbb{SE}(3)$ at $T \in \mathbb{SE}(3)$ is the unique matrix $\mathrm{Ad}_T \in \mathbb{R}^{6\times 6}$ such that

$$T\,\mathrm{Exp}(\xi) = \mathrm{Exp}(\mathrm{Ad}_T\,\xi)T \tag{4.36}$$

for all $\xi \in \mathbb{R}^6$. Again, the analytic expression is available in [155, Appendix].

4.1.8.2 Proof of Theorem 4.1

Before we prove Theorem 4.1, we derive a separating hyperplane result, Theorem 4.4. It defines the hyperplane that separates potential obstacle points from the free space after an uncertain rigid transformation.

Lemma 4.4. *Let the transform between two frames be $T_A^B \sim \mathcal{N}(\widehat{T}_A^B, \Sigma)$. Consider a point*

$p|^A \in \mathbb{R}^3$. *Given Assumption 4.1, for any non-zero vector* $a \in \mathbb{R}^3$,

$$p|^B = T_A^B \cdot p|^A \in \mathcal{H} \tag{4.37}$$

where

$$\mathcal{H} = \{p \in \mathbb{R}^3 : a^T p \geq r\} \tag{4.38a}$$

$$r = a^T(\widehat{T}_A^B \cdot p|^A) - \sqrt{a^T \Sigma_p a} \tag{4.38b}$$

and $\Sigma_p \in \mathbb{S}^3_{++}$ *is as defined by Assumption 4.1.*

Proof of Theorem 4.4. By Assumption 4.1, the transformed point satisfies

$$p|^B \in \mathcal{E} = \left\{p \in \mathbb{R}^3 : \left\|\Sigma_p^{-1/2}(p - \hat{p})\right\| \leq 1\right\}$$

where $\hat{p} = \widehat{T}_A^B \cdot p|^A$, and $\Sigma_p \in \mathbb{S}^3_{++}$ is defined in Assumption 4.1. Next, we define

$$p^\perp = \hat{p} - \frac{\Sigma_p a}{\sqrt{a^T \Sigma_p a}}$$

such that $p^\perp \in \mathbb{R}^3$ is on the surface of the ellipsoid and has a surface normal $-a$. Therefore, the set of points $\mathcal{H} = \{p \in \mathbb{R}^3 : a^T(p - p^\perp) \geq 0\}$ contains the ellipsoid, i.e., $\mathcal{E} \subset \mathcal{H}$,

$$r = a^T p^\perp = a^T \hat{p} - \frac{a^T \Sigma_p a}{\sqrt{a^T \Sigma_p a}} = a^T \hat{p} - \sqrt{a^T \Sigma_p a}$$

which completes the proof. □

We can now prove Theorem 4.1.

Proof of Theorem 4.1. It suffices to show that any obstacle potentially on the boundary of $\mathcal{P}_k$ will not be in $\mathcal{P}_{k+1}$. Consider an obstacle point $o|^{B_k} = p|^{B_k} + \epsilon a_k$, where $\epsilon > 0$ and $p|^{B_k}$ is a point on the surface of $\mathcal{P}_k$. Then for some $i \in \{1, ..., N\}$,

$$a_{k,i}^T p|^{B_k} = b_{k,i}.$$

After the rigid transformation, by Theorem 4.4, $o|^{B_{k+1}} \in \mathcal{E} \subset \{p : a_{k+1,i}^T p \geq r\}$ where

$$\begin{aligned}
r &= a_{k+1,i}^T(\widehat{T}_{B_k}^{B_{k+1}} \cdot o|^{B_k}) - \sqrt{a_{k+1,i}^T \Sigma_p a_{k+1,i}} \\
&= a_{k+1,i}^T(R(p|^{B_k} + \epsilon a_{k,i}) + t) - \sqrt{a_{k+1,i}^T \Sigma_p a_{k+1,i}} \\
&= a_{k,i}^T(p|^{B_k} + \epsilon a_{k,i}) + a_{k,i}^T R^T t - \sqrt{a_{k+1,i}^T \Sigma_p a_{k+1,i}} \\
&= b_{k,i} + \epsilon \left\|a_{k,i}\right\|^2 + a_{k,i}^T R^T t - \sqrt{a_{k+1,i}^T \Sigma_p a_{k+1,i}} \\
&= b_{k+1,i} + \epsilon \left\|a_{k,i}\right\|^2 + \rho_i - \sqrt{a_{k+1,i}^T \Sigma_p a_{k+1,i}}
\end{aligned}$$

Now consider the last term:

$$\begin{aligned}
\sqrt{a_{k,i+1}^T \Sigma_p a_{k+1,i}} &= \left\|\Sigma_p^{1/2} a_{k+1,i}\right\| \\
&= \left\|\sqrt{\kappa}\Sigma_k^{1/2} J^T a_{k+1,i}\right\| \\
&= \left\|\sqrt{\kappa}\Sigma_k^{1/2} \begin{bmatrix} R^T \\ -(R[o|^{B_k}]_\times)^T \end{bmatrix} a_{k+1,i}\right\| \\
&= \left\|\sqrt{\kappa}\Sigma_k^{1/2} \begin{bmatrix} a_{k,i} \\ [o|^{B_k}]_\times a_{k,i} \end{bmatrix}\right\| \\
&= \left\|\sqrt{\kappa}\Sigma_k^{1/2} \begin{bmatrix} a_{k,i} \\ -[a_{k,i}]_\times o|^{B_k} \end{bmatrix}\right\| \\
&= \left\|\sqrt{\kappa}\Sigma_k^{1/2} \begin{bmatrix} a_{k,i} \\ -[a_{k,i}]_\times p|^{B_k} \end{bmatrix}\right\|
\end{aligned}$$

where in the last line, we used $[a_{k,i}]_\times(\epsilon a_{k,i}) = 0$.

Finally, since Σ_k is positive definite, this expression is convex wrt $p|^{B_k}$. Considering $p|^{B_k}$ must be some convex combination of the vertices on the i-th face,

$$\begin{aligned}
\left\|\Sigma_p^{1/2} a_{k+1,i}\right\| &\leq \max_{j \in \{1,\ldots,m_i\}} \left\|\sqrt{\kappa}\Sigma_k^{1/2} \begin{bmatrix} a_{k,i} \\ -[a_{k,i}]_\times v_{i,j}|^{B_k} \end{bmatrix}\right\| \\
&= \rho_i
\end{aligned}$$

where $v_{i,j}|^{B_k}$ is the j-th vertex on the i-th face of $\mathcal{P}_k$.

Therefore, we have

$$\begin{aligned} r &= b_{k+1,i} + \epsilon \left\| a_{k,i} \right\|^2 + \rho_i - \left\| \Sigma_p^{1/2} a_{k+1,i} \right\| \\ &\geq b_{k+1,i} + \epsilon \left\| a_{k,i} \right\|^2 > b_{k+1,i}, \end{aligned}$$

that is,

$$\begin{aligned} & o|^{B_{k+1}} \in \mathcal{E} \subset \{p : a_{k+1,i}^T p \geq r\}, \\ \implies & o|^{B_{k+1}} \notin \{p : a_{k+1,i}^T p \leq b_{k+1,i}\} \end{aligned}$$

which completes the proof. □

4.1.8.3 Proof of Theorem 4.3

Proof. Consider any point $p|^{B_{k+1}}$. When represented in frame B_k, it could correspond to a set of points within the ellipsoid

$$p|^{B_k} \in \mathcal{E} = \{p \in \mathbb{R}^3 : \left\| \Sigma_p^{-1/2} (p - \hat{p}) \right\| \leq 1\}$$

where $\hat{p} = \widehat{T}_{B_{k+1}}^{B_k} \cdot p|^{B_{k+1}}$, and $\Sigma_p \in \mathbb{S}_{++}^3$ is as defined by Assumption 4.1. Therefore,

$$\begin{aligned} d(p|^{B_{k+1}}) &\overset{(1)}{\geq} \min_{p|^{B_k} \in \mathcal{E}} d(p|^{B_k}) \\ &\overset{(2)}{\geq} \min_{p|^{B_k} \in \mathcal{E}} d_M^k(\widehat{T}_{B_k}^M \cdot p|^{B_k}) \\ &\overset{(3)}{\geq} d_M^k(\widehat{T}_{B_k}^M \cdot \hat{p}) - \operatorname{diam}(\mathcal{E})/2 \\ &\overset{(4)}{=} d_M^k(\widehat{T}_{B_k}^M \widehat{T}_{B_{k+1}}^{B_k} \cdot p|^{B_{k+1}}) - \sqrt{\lambda_{\max}(\Sigma_p)} \\ &\overset{(5)}{=} d_M^k(\widehat{T}_{B_{k+1}}^M \cdot p|^{B_{k+1}}) - \sqrt{\lambda_{\max}(\Sigma_p)} \\ &\overset{(6)}{=} d_M^{k+1}(\widehat{T}_{B_{k+1}}^M \cdot p|^{B_{k+1}}) \end{aligned}$$

where $\operatorname{diam}(\mathcal{E})$ is the diameter of $\mathcal{E}$. (1) is true by defition, (2) uses the fact that d_M^k is a certified-ESDF. (3) is true because ESDFs have unit gradient everywhere, (4) uses the eigenvalue of Σ_p to bound the ellipsoid with a sphere, and (5), and (6) are basic simplifications. Therefore, d_M^{k+1} is also a certified-ESDF. □

4.1.8.4 Extracting Covariance of Relative Transforms from Odometry with Covariance

To the best of the author's knowledge, all Visual Odometry (VO)/VIO/SLAM algorithms report the mean odometry estimate and the covariance with respect to the initial frame: at the k-th frame, the following quantities are available:

$$\widehat{T}_{B_k}^{B_0} \in \mathbb{SE}(3), \quad \Sigma_{B_k}^{B_0} \in \mathbb{S}_{++}^6 \tag{4.39}$$

i.e., the pose of the k-th body frame with respect to the initial frame, and the covariance of the estimate.

However, to use the frameworks proposed in this section, the relative transform and its covariance are required:

$$\widehat{T}_{B_{k+1}}^{B_k} \in \mathbb{SE}(3), \quad \Sigma_{B_{k+1}}^{B_k} \in \mathbb{S}_{++}^6. \tag{4.40}$$

Here we detail a method to obtain these quantities.

Consider the following result adapted from [111, Section VIII] to match the convention used in this section.

Lemma 4.5. *Let $T_{ij}, T_{ik}, T_{jk} \in \mathbb{SE}(3)$ represent the poses between coordinate frames $(i,j),(i,k)$, and (j,k) respectively. Let $\hat{T}_{\cdot}$ be the corresponding estimated transform. Let*

$$T_{ij} = \hat{T}_{ij} \operatorname{Exp}(\xi_{ij}) \tag{4.41}$$

and similar for $(ik),(jk)$. Suppose

$$\begin{bmatrix} \xi_{ij} \\ \xi_{ik} \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} \Sigma_{ij} & \Sigma_{ij,jk} \\ \Sigma_{ij,ik}^T & \Sigma_{ik} \end{bmatrix} \right). \tag{4.42}$$

Then, the estimated relative transform is

$$\hat{T}_{jk} = \hat{T}_{ij}^{-1} \hat{T}_{ik} \tag{4.43}$$

and the associated covariance is (to first order)

$$\Sigma_{jk} = A\Sigma_{ij}A^T + \Sigma_{ik} - A\Sigma_{ij,ik} - \Sigma_{ij,jk}^T A^T, \tag{4.44}$$

where $A = \operatorname{Ad}_{\hat{T}_{jk}^{-1}} \in \mathbb{R}^{6\times 6}$ is the adjoint matrix of $\mathbb{SE}(3)$ at $\hat{T}_{jk}^{-1}$.

Notice that the negative signs on the cross terms implies that a non-zero $\Sigma_{ij,jk}$ decreases the covariance of the relative pose.

Proof. Since $T_{jk} = T_{ij}^{-1} T_{ik}$, the following must hold:

$$\begin{aligned}
\hat{T}_{jk} \operatorname{Exp}(\xi_{jk}) &= \left(\hat{T}_{ij} \operatorname{Exp}(\xi_{ij})\right)^{-1} \left(\hat{T}_{ik} \operatorname{Exp}(\xi_{ik})\right) \\
&= \operatorname{Exp}(-\xi_{ij}) \hat{T}_{ij}^{-1} \hat{T}_{ik} \operatorname{Exp}(\xi_{ik}) \\
&= \operatorname{Exp}(-\xi_{ij}) \hat{T}_{jk} \operatorname{Exp}(\xi_{ik}) \\
&= \hat{T}_{jk} \operatorname{Exp}(-\operatorname{Ad}_{\hat{T}_{jk}^{-1}} \xi_{ij}) \operatorname{Exp}(\xi_{ik})
\end{aligned}$$

where in the last equality we used the following property of the adjoint matrix: $\operatorname{Exp}(\xi) T = T \operatorname{Exp}(\operatorname{Ad}_{T^{-1}} \xi)$ for any $T \in \mathbb{SE}(3)$ and $\xi \in \mathbb{R}^6$.

Defining $\xi'_{ij} = -\operatorname{Ad}_{\hat{T}_{jk}^{-1}} \xi_{ij}$, we have

$$\operatorname{Exp}(\xi_{jk}) = \operatorname{Exp}(\xi'_{ij}) \operatorname{Exp}(\xi_{ik})$$

and therefore using the Baker-Campbell-Hausdorff (BCH) formula (see [111]), the first order estimated covariance is

$$\begin{aligned}
E[\xi_{jk} \xi_{jk}^T] &\approx \underbrace{E[\xi'_{ij} {\xi'_{ij}}^T] + E[\xi_{ik} \xi_{ik}^T]}_{\text{2nd order diag. terms}} \\
&\quad + \underbrace{E[\xi'_{ij} \xi_{ik}^T] + E[\xi_{ik} {\xi'_{ik}}^T]}_{\text{2nd order cross terms}} \\
&= A \Sigma_{ij} A^T + \Sigma_{ik} - A \Sigma_{ij,ik} - \Sigma_{ij,jk}^T A^T
\end{aligned}$$

where $A = \operatorname{Ad}_{\hat{T}_{jk}^{-1}}$. This completes the proof. □

We can now apply this lemma to estimate the relative transforms between successive frames. Recall the odometry algorithm defines the covariances as

$$T_{B_k}^{B_0} = \widehat{T}_{B_k}^{B_0} \operatorname{Exp}(\xi_{k,0}), \quad \xi_{k,0} \sim \mathcal{N}(0, \Sigma_{k,0}) \tag{4.45}$$

and similar for $k+1$. The perturbations ξ are assumed to be correlated,

$$\begin{bmatrix} \xi_{k,0} \\ \xi_{k+1,0} \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} \Sigma_{k,0} & \Sigma_{k,0;k+1,0} \\ * & \Sigma_{k+1,0} \end{bmatrix} \right) \tag{4.46}$$

where the $*$ indicates to the symmetric element.

We assume that the two poses are highly correlated, with a correlation coefficient $\rho \in [-1, 1]$, (we chose $\rho = 0.99$). Then,

$$\Sigma_{k,0;k+1,0} = \rho \left(\Sigma_{k,0} \Sigma_{k+1,0}^T \right)^{1/2} \tag{4.47}$$

Then, using Theorem 4.5, the estimated relative transform is

$$\widehat{T}_{B_{k+1}}^{B_k} = (\widehat{T}_{B_k}^{B_0})^{-1} \widehat{T}_{B_{k+1}}^{B_0} \tag{4.48}$$

and the estimated relative covariance is

$$\Sigma_{B_{k+1}}^{B_k} = A \Sigma_{B_k}^{B_0} A^T + \Sigma_{B_{k+1}}^{B_0} - A \Sigma_\times - \Sigma_\times^T A^T \tag{4.49}$$

where

$$\Sigma_\times = \rho \left(\Sigma_{B_k}^{B_0} (\Sigma_{B_{k+1}}^{B_0})^T \right)^{1/2}, \quad A = \mathrm{Ad}_{(\widehat{T}_{B_{k+1}}^{B_k})^{-1}}.$$

Note, the adjoint matrix for $T = \begin{bmatrix} R & t \\ 0 & 1 \end{bmatrix} \in \mathbb{SE}(3)$ is

$$\mathrm{Ad}_T = \begin{bmatrix} R & [t]_\times R \\ 0 & R \end{bmatrix}$$

and $\mathrm{Ad}_{T^{-1}} = (\mathrm{Ad}_T)^{-1}$ [155].

4.1.8.5 Replica Dataset Environment Details

Table 4.4 shows the size and volume of the bounding box for each environment used in the simulation studies. It also shows the number of mesh points in the environment.

Table 4.4: Size and volume of each environment used.

Env.	Length X (m)	Length Y (m)	Length Z (m)	Bounding Box Volume (m^3)	Number of Mesh Points
`office0`	4.40	5.01	2.99	65.95	589 517
`office1`	4.81	4.11	2.80	55.24	423 007
`office2`	6.47	8.14	2.77	145.89	858 623
`office3`	8.64	9.20	3.10	246.85	1 187 140
`office4`	6.55	6.51	2.82	119.96	993 008
`room0`	7.76	4.70	2.81	102.43	954 492
`room1`	6.65	5.73	2.75	104.81	645 512
`room2`	6.77	4.95	3.59	120.34	722 496

4.1.8.6 Additional Simulation Results

Table 4.5 and Table 4.6 show additional results of the performance of the SFC and ESDF methods on the Replica dataset. Here we show the results from a trajectory perturbed by Σ =1e-5I and Σ =1e-6I.

Table 4.5: Results of the three Safe Flight Corridor (SFC) methods on the Replica dataset. Each environment was run with $\Sigma = \sigma^2 I$ for two different σ^2 values, 1e-5 and 1e-6.

		Violation Rate (%)			Max Violation (mm)			SFC Volume (m^3)		
Env	σ^2	Baseline	Heuristic	Certified	Baseline	Heuristic	Certified	Baseline	Heuristic	Certified
office0	1e-6	18.6%	0.1%	0.0%	102.74	22.05	0.03	34.8	6.7	5.7
	1e-5	32.5%	0.6%	0.0%	397.89	33.83	0.03	38.9	6.8	4.8
office1	1e-6	12.8%	0.6%	0.0%	95.30	14.48	0.86	17.6	3.6	2.6
	1e-5	12.9%	0.1%	0.0%	373.39	24.65	0.86	17.7	3.7	2.0
office2	1e-6	10.1%	0.1%	0.0%	159.66	18.42	0.39	40.8	4.3	3.6
	1e-5	21.3%	0.9%	0.0%	299.11	21.93	0.39	44.9	4.3	3.0
office3	1e-6	12.7%	0.1%	0.0%	177.65	11.61	0.88	56.6	4.6	3.0
	1e-5	16.5%	0.0%	0.0%	460.25	7.38	0.94	57.9	4.6	0.9
office4	1e-6	14.4%	0.3%	0.0%	125.48	8.91	1.69	63.3	15.7	12.5
	1e-5	24.6%	4.7%	0.0%	262.23	82.75	1.69	66.5	16.1	10.6
room0	1e-6	10.7%	0.0%	0.0%	117.12	11.02	0.95	53.0	12.3	9.1
	1e-5	20.1%	0.5%	0.0%	396.74	47.97	0.95	55.8	12.3	8.0
room1	1e-6	19.2%	0.4%	0.0%	191.43	14.20	0.71	38.7	6.9	5.8
	1e-5	25.7%	1.1%	0.0%	377.01	23.68	0.71	39.5	6.7	5.3
room2	1e-6	6.8%	0.9%	0.0%	85.02	12.85	0.65	29.4	7.5	4.4
	1e-5	11.1%	1.5%	0.0%	322.36	25.63	0.65	30.1	7.5	1.8

4.1.8.7 Effect of Odometry Covariance

4.1.8.8 Effect of the Deflation on the Volume of Certified Free Space

Due to the deflation of the free space in the certified methods, the volume of space that a path planner can use to navigate the robot will be smaller than in the baseline methods.. In this section, we show qualitatively and quantitatively the volume of free space usable by a robotic system.

The rover was operated in a room approximately 40 × 20 m large drawn in Figure 4.8. Starting in the middle, the robot was teleoperated to explore and map the room. The robot has a horizontal field of view of 75°, and a maximum depth integration distance of 8 m. This means that from the depth image, the maximum distance that NvBlox will mark as free or safe is 8 m from the camera origin. Thus, in these experiments, the heuristic method also uses a forgetting radius of 8 m.

A quantitative comparison of the algorithms is presented in Figure 4.9a, b. In (a) we can see the area of the claimed safe region by each of the three methods. Although the claimed

Table 4.6: Results of the three Euclidean Signed Distance Field (ESDF) methods on the Replica dataset.

		Violation Rate (%)			Max Violation (mm)			ESDF Volume (m^3)		
Env	σ^2	Baseline	Heuristic	Certified	Baseline	Heuristic	Certified	Baseline	Heuristic	Certified
office0	1e-6	48.1%	31.6%	0.5%	604.3	563.6	109.5	46.1	39.5	10.7
	1e-5	21.2%	11.8%	0.5%	384.2	322.5	107.7	42.3	38.1	10.9
office1	1e-6	35.3%	34.4%	0.1%	406.9	379.5	82.5	23.2	23.0	3.8
	1e-5	11.1%	10.6%	0.3%	172.0	172.0	93.8	21.9	21.8	4.2
office2	1e-6	51.5%	7.6%	0.1%	520.0	311.8	141.4	77.5	31.3	6.2
	1e-5	23.8%	2.0%	0.1%	212.6	253.8	100.0	68.7	31.1	6.2
office3	1e-6	54.7%	4.7%	0.0%	671.1	429.4	100.0	110.9	42.0	5.0
	1e-5	28.2%	1.5%	0.0%	330.5	226.3	72.1	96.9	41.4	6.0
office4	1e-6	48.3%	10.1%	0.1%	636.9	366.6	66.3	99.7	51.5	14.3
	1e-5	21.0%	3.9%	0.1%	260.0	215.4	69.3	90.9	50.9	14.4
room0	1e-6	62.0%	9.2%	2.4%	990.8	428.5	120.0	105.4	28.6	31.5
	1e-5	34.4%	3.2%	3.2%	335.3	244.1	164.9	90.9	27.6	32.9
room1	1e-6	48.1%	20.9%	0.0%	604.6	384.7	100.0	53.8	34.5	6.6
	1e-5	17.5%	8.8%	0.0%	240.0	169.7	72.1	47.6	33.1	6.9
room2	1e-6	47.5%	16.3%	0.1%	594.0	435.4	82.5	63.6	38.7	4.5
	1e-5	21.9%	5.1%	0.0%	291.9	200.0	66.3	56.8	37.8	9.5

Table 4.7: Performance of the three Euclidean Signed Distance Field (ESDF) methods in the Office0 environment under varying odometry covariance at $\kappa = 3$.

	Violation Rate (%)			Max Violation (mm)			ESDF Volume (m^3)		
σ^2	Baseline	Heuristic	Certified	Baseline	Heuristic	Certified	Baseline	Heuristic	Certified
1e-04	61.09%	43.46%	0.19%	1.24	1.24	0.17	63.87	45.37	10.68
1e-05	48.15%	31.55%	0.50%	0.60	0.56	0.11	46.14	39.46	10.74
1e-06	21.16%	11.79%	0.49%	0.38	0.32	0.09	42.33	38.11	10.94
1e-07	5.17%	2.75%	0.49%	0.22	0.19	0.10	41.79	37.93	11.59
1e-08	2.04%	1.72%	0.54%	0.18	0.13	0.13	41.70	37.86	16.54
1e-09	1.94%	1.54%	0.62%	0.18	0.12	0.16	41.69	37.85	31.52
1e-10	1.91%	1.51%	0.77%	0.18	0.11	0.16	41.69	37.85	37.18
1e-11	1.93%	1.53%	1.21%	0.18	0.11	0.16	41.69	37.86	41.20
1e-12	1.93%	1.53%	1.35%	0.18	0.11	0.16	41.69	37.86	41.59

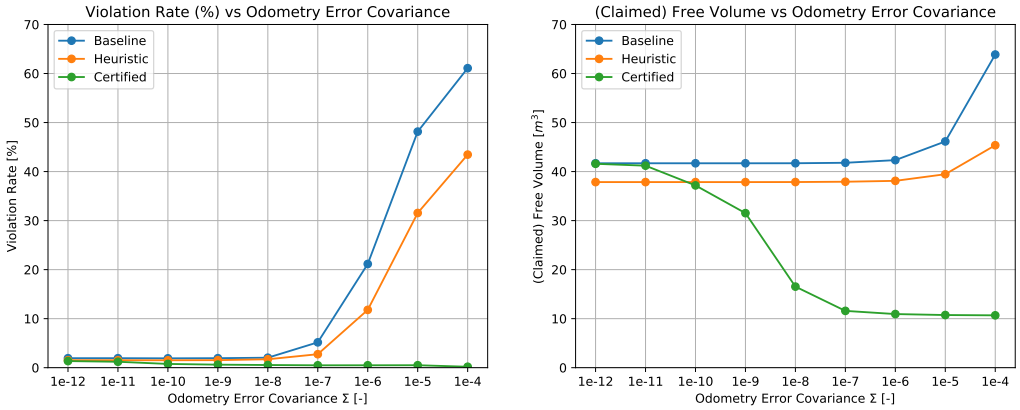

Figure 4.7: Plots of the effect of the odometry covariance on the performance of the three SDF methods. On the left, we can see that as the odometry covariance increases, the maximum violation rate increases, except for the certified method. The right plot shows that the volume of claimed free space (incorrectly) increases with odometry covariance, while it (correctly) decreases with the certified method.

free region is largest for the baseline method, as discussed below, the map is erroneous. The certified and heuristic methods have similar free area, although the heuristic method is also often incorrect.

In Figure 4.9b, we show the distance to the furthermost safe point from the robot position. This gives an indication of extent of the map that would be free if it were not for the obstacles in the environment. Here, we can see that compared to the maximum integration distance of 8 m, the certified method has its furthermost safe voxel approximately 12 m away, and upto 18 m away. In contrast, the heuristic method is clipped at 8 m. The evolution of the maps in time is clearer in the accompanying video, where the FPV and third person view of the robot are also drawn.

Slices of the ESDF and the Certified ESDF are shown in Figure 4.9c, d. The robot's trajectory is also drawn. Compare Figure 4.9c1-c4. We can see that the map drifts significantly - in (c1) we use a gray dashed line to highlight the end of the corridor as mapped at that time. In (c4), we draw the corridor mapped in (c1) as well as the newly mapped corridor, and we can see a significant shift in the map. In (d1-d4) we can see the certified ESDF region marked in green, and even as the robot moves around a significant part of the area around the robot remains part of the safe region.

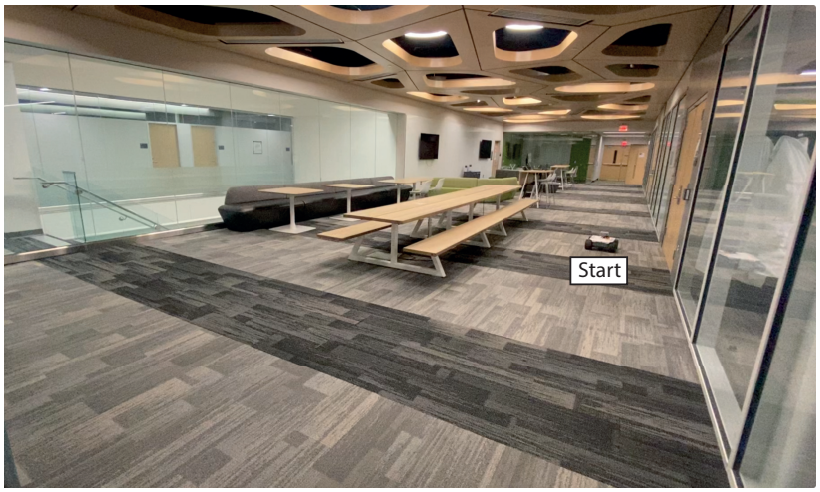

Figure 4.8: Experimental domain used in Figure 4.9.

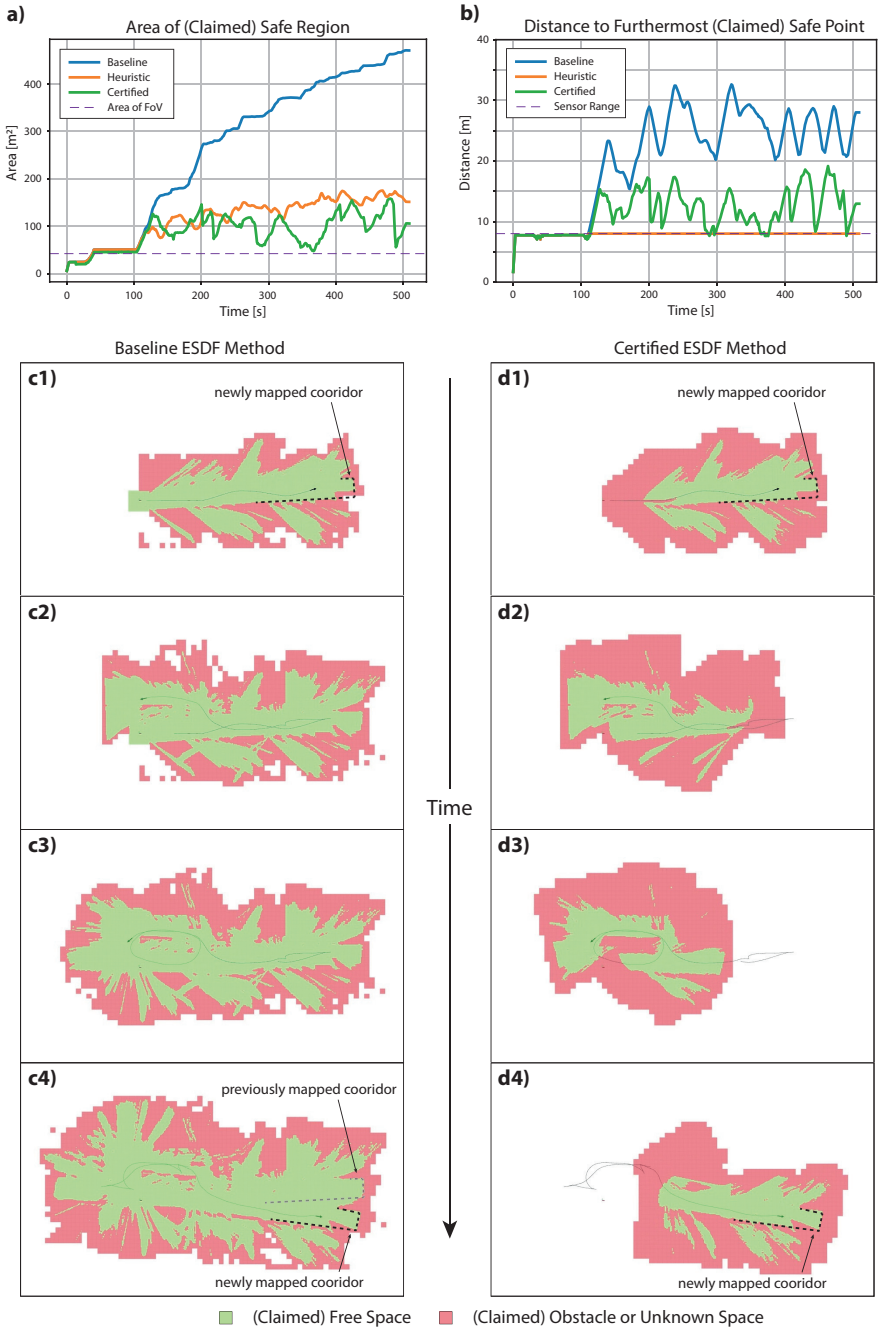

Figure 4.9: Quantitative and qualitative analysis of the effect of the deflation on the volume of the certified free space. (a) Compares the area of the claimed safe region on a 2D slice of the ESDF extracted at the robot height. As a reference, the area of the FoV of the camera is also drawn. (b) Compares the distance of the furthermost (claimed) free voxel from the robot position. As a reference, the maximum depth of the depth sensor (8 m) is indicated. In (c1-c4) we see snapshots of the map generated by the Baseline ESDF method, and in (d1-d4) we see the corresponding snapshots from the Certified ESDF method. The accompanying video animates the map slices and is therefore clearer.

CHAPTER 5

Information Gathering and Perceivability

The previous chapters have developed architectures for safety critical control planning and perception. At this point we consider carefully the interaction of the constrained controllers with the mission-level objectives. As a canonical example, we consider the case of a team of robots exploring an environment with the objective of collecting information. This information can be useful either to satisfy the constraints directly, or simply part of the mission objective.

In this chapter, we formalize the notion of information collection and try to understand the connection between the robot, the environment and the information collected. At a fundamental level, we seek to answer whether a robotic system even has the ability to acquire certain information. For example, certain linear systems are not observable and some are not controllable, so no matter the algorithm designed there simply will be no way to estimate the state or control the system respectively. Here, we attempt to quantify a similar property for information gathering, and we term it the perceivability of the environment.

To develop this notion, we must also introduce a metric of information, which we term clarity. Clarity is equivalent to differential entropy, but has certain convenient properties for the informative path planning problem. We have also used clarity and the clarity dynamics of a Kalman Filter to effectively design multiagent dynamic coverage controllers.

5.1 Clarity and Perceivability: Fundamental Limits of Information Gathering

Robots are often deployed to explore unknown or unstructured environments, e.g., ocean gliders collecting oceanographic data, or aerial robots searching for targets in a disaster response. In this section, we establish two concepts: *clarity* and *perceivability*, to capture information acquisition and their use in the design of informative controllers.

Informative Path Planning (IPP) seeks to design trajectories that maximize the 'amount of information' collected subject to budgetary constraints such as total energy or time [79]. 'Information' is measured in many ways, e.g. entropy/mutual information [43, 58, 121, 165], Fisher Information [185], the number of unexplored cells/frontiers or the area of Voronoi partitions [42, 52, 86, 187], Gaussian Processes [112, 131], and data-informativity [166]. Various techniques to solve IPP exist, including grid/graph-search or sampling [39, 43, 121, 175]. While useful for trajectory generation, such methods cannot quantify whether information can be gathered in the first place.

The main objective is to answer the following questions: *Given a platform (e.g., a robot) with onboard sensors, and an environment in which information is to be collected, (1) does the overall system have sufficient actuation and sensing capabilities to gather information in a specified time, and (2) what are optimal control strategies to collect the information?*

To address these, we first introduce *clarity* as a measure of the quality of information possessed. Clarity about a random variable m, denoted $q[m]$, lies in $[0, 1]$, where $q = 0$ corresponds to the case where m is completely unknown, and $q = 1$ to the case where m is perfectly known in an idealized (noise-free) setting. Clarity is inspired by differential entropy, but compared to the latter it takes finite values with finite time derivatives. As a first contribution, we show that if m is estimated using a Kalman Filter, the rate of change of clarity has a similar structure to one assumed in dynamic coverage controllers [28, 77, 129]. This establishes certain optimality properties for dynamic coverage control, rather than being viewed as a heuristic for exploration.

The second and primary contribution is the definition of *perceivability*, which quantifies the maximum achievable clarity about the environment in a fixed time by a given system (robot dynamics and sensory outputs). It depends on the controllability of the system describing the robot dynamics, on the observability of the system describing the environment's evolution. This coupling makes perceivability distinct from standard notions of controllability (whether the robot state can be driven to a desired state) or observability (whether the robot state can be uniquely determined from sensory outputs).

We show that perceivability is linked to reachability analysis of an augmented system

including both the robot's system dynamics, and the environment's clarity dynamics. We show that perceivability can be determined by solving a Hamilton-Jacobi-Bellman (HJB) equation, which allows us to determine optimal controllers, i.e., those that maximize the quality of information acquired about an environment.

In Sec. 5.1.1 and 5.1.2 we introduce clarity and perceivability, respectively, and in Sec. 5.1.3 we demonstrate these ideas. Background material is presented where needed.

5.1.1 Clarity

To aid the reader, we use the following running example, inspired by an oceanographic mission: we wish to create a map of the ocean-surface temperature using sensors onboard a surface vessel, or thermal images from an aerial vehicle, both subject to ocean currents or winds. We require a suitable information metric: for this, we propose clarity.

5.1.1.1 Definitions and Fundamental Properties

Definition 5.1. *[160, Ch. 8] X is a **continuous random variable** if its **cumulative distribution** $F(x) = Pr(X \leq x)$ is continuous. The **probability density function** is $f(x) = F'(x)$. The set where $f(x) > 0$ is the **support set** of X.*

Differential entropy extends the notion of entropy [149] from discrete to continuous random variables:

Definition 5.2. *[160, Ch. 8] The **differential entropy** $h[X]$ of a continuous random variable X with density $f(x)$ is*

$$h[X] = -\int_S f(x) \log f(x) dx \tag{5.1}$$

where S is the support set of X.

While differential entropy shares many properties with discrete entropy [160, Sec. 2.1], there are key differences. E.g., while discrete entropy is non-negative, differential entropy is in $[-\infty, \infty]$, i.e., *it can be negative*. We define *clarity* as:

Definition 5.3. *Let X be a n-dimensional continuous random variable with differential entropy $h[X]$. The **clarity** of X is*

$$q[X] = \left(1 + \frac{\exp\left(2h[X]\right)}{(2\pi e)^n}\right)^{-1}. \tag{5.2}$$

The normalizing factor $(2\pi e)^n$ is introduced to simplify some of the algebra, as in Theorem 5.1 and the next example:

Example 5.1. Consider $X \sim \mathcal{U}(a, b)$, and $Y \sim \mathcal{N}(\mu, P)$, where $a, b \in \mathbb{R}$, $\mu \in \mathbb{R}^n$, $P \in \mathbb{S}^n_+$. Then,

$$h[X] = \log(b-a), \quad h[Y] = \log\sqrt{(2\pi e)^n |P|},$$
$$q[X] = \frac{1}{1+\frac{(b-a)^2}{2\pi e}}, \quad q[Y] = \frac{1}{1+|P|}.$$

Next, we establish some fundamental properties of clarity.

Property 5.1. *For any n-dimensional continuous random variable X, $A \in \mathbb{R}^{n\times n}$, and $c \in \mathbb{R}^n$,*

$$q[X] \in [0,1] \qquad \textit{(clarity is bounded)} \tag{5.3}$$
$$q[X+c] = q[X] \qquad \textit{(clarity is shift-invariant)} \tag{5.4}$$
$$q[AX] \neq q[X] \qquad \textit{(clarity is not scale-invariant)} \tag{5.5}$$

Proof. Of (5.3)*:* Since $h[X] \in [-\infty, \infty]$, $q[X] = 1/(1+s)$ for some $s \in [0, \infty]$, i.e., $q[X] \in [0, 1]$.

Of (5.4)*,* (5.5)*:* Follows from [160, Th. 8.6.3] ($h[X+c] = h[X]$), and [160, Th. 8.6.4] ($h[AX] = h[X] + \log|A|$). □

In information gathering tasks, we seek to design trajectories that minimize the estimation error. Let X be a random variable of *any* distribution with clarity $q[X]$. Let $\hat{X}$ be *any* estimate of X, then $E[(X-\hat{X})(X-\hat{X})^T]$ is the expected estimation error.[1] Theorem 5.1 shows for expected estimation error to approach 0 it is necessary that clarity approach 1.

Theorem 5.1. *For any n-dimensional continuous random variable X and any $\hat{X} \in \mathbb{R}^n$, the determinant of the expected estimation error is lower-bounded as*

$$\left|E[(X-\hat{X})(X-\hat{X})^T]\right| \geq \frac{1}{q[X]} - 1, \tag{5.6}$$

with equality if and only if X is Gaussian and $\hat{X} = E[X]$.

[1] This is the covariance of X only if $\hat{X}$ is the mean of X. Since we do not make this assumption, we refer to this quantity as the expected estimation error.

Proof. Following the same arguments as in [160, Th. 8.6.6],

$$\begin{aligned}\left|E[(X-\hat{X})(X-\hat{X})^T]\right| &\geq \min_{\hat{X}\in\mathbb{R}^n}\left|E[(X-\hat{X})(X-\hat{X})^T]\right| \\ &= \left|E[(X-E[X])(X-E[X])^T]\right| \\ &= |\mathrm{var}(X)|\end{aligned}$$

and since a Guassian distribution has the greatest entropy of a given variance [160, Th. 8.6.6],

$$\left|E[(X-\hat{X})(X-\hat{X})^T]\right| \geq \frac{e^{2h[X]}}{(2\pi e)^n} = \frac{1}{q[X]} - 1. \qquad \square$$

Corollary 5.2. *For any 1-D continuous random variable x and any $\hat{x} \in \mathbb{R}$, the expected estimation error is lower-bounded as*

$$E[(x-\hat{x})^2] \geq \frac{1}{q[x]} - 1 \tag{5.7}$$

with equality if and only if x is Gaussian and $\hat{x} = E[x]$.

Proof. Use Thm. 5.1 with $P \in \mathbb{S}^1_{++} \implies |P| = P$. $\square$

Remark 5.1. *Although there is a one-to-one mapping between clarity and differential entropy* (5.2)*, the primary benefits of clarity are: (I) clarity is bounded over* $[0,1]$ *instead of* $[-\infty,\infty]$*, (II) the time derivatives, defined later in* (5.12)*, are finite for all* $q \in [0,1]$*. This is particularly important for perceivability, since numerical methods to solve the HJB equation, defined later in* (5.20)*, require bounded values and derivatives.*

5.1.1.2 Connection between Clarity and Coverage Control

Consider the system

$$\dot{x} = f(x,u) \tag{5.8}$$

where the state is $x \in \mathcal{X} \subset \mathbb{R}^n$, control input is $u \in \mathcal{U} \subset \mathbb{R}^m$.

The objective in coverage control is to design a controller $\pi : \mathcal{X} \to \mathcal{U}$ for the system (5.8) such that closed-loop trajectories gather information over a domain $\mathcal{D} \subset \mathcal{X}$. As in [77], let $c = c(t,p)$ denote the 'coverage level' about a point $p \in \mathcal{D}$ at time t. [77] assumes the coverage increases through a sensing function $S : \mathcal{X} \times \mathcal{D} \to \mathbb{R}_{\geq 0}$ (positive when p can be sensed from x, and 0 else), and coverage decreases at a rate $\alpha : \mathcal{D} \to \mathbb{R}_{\geq 0}$. This results in

the model

$$\dot{c} = S(x,p)(1-c) - \alpha(p)c. \tag{5.9}$$

In [28, 129] the α term is ignored, and a point p is said to be 'covered' if $c(t,p)$ reaches a threshold c^*.

However, given specifications on the robot, sensors, and the environment, it is not clear how to systematically define S, α, c^*. [28, 77, 129] resort to heuristic methods.

In many practical scenarios, measurements are assimilated using a Kalman Filter. In principle, the coverage dynamics should reflect the information gathering mechanism, i.e., the evolution of the quality of information about the environment as it is estimated using the Kalman Filter. In deriving the clarity dynamics, (see (5.12)), we notice similarities with (5.9).

Consider the simplest scenario, where we want to estimate a scalar variable $m \in \mathbb{R}$. We assume m is a stochastic process:

$$\dot{m} = w(t), \qquad w(t) \sim \mathcal{N}(0, Q), \tag{5.10}$$

$$y = C(x)m + v(t), \qquad v(t) \sim \mathcal{N}(0, R(x)), \tag{5.11}$$

where $y \in \mathbb{R}$ is the measurement. Notice $C(x), R(x)$ are robot-state dependent, emphasizing that the quality of the measurements can depend on the robot's state. For simplicity, assume x is known. The following demonstrates the setup:

Example 5.2. Let x be the quadrotor's state, with position $x_{pos} \in \mathbb{R}^2$ and altitude x_{alt}. The quadrotor uses a downward facing thermal camera with half-cone angle θ to measure the ocean's temperature m at a location p. Then $C(x)$ is

$$C(x) = \begin{cases} 1, & \text{if } \|x_{pos} - p\| \leq x_{alt} \tan\theta, \\ 0, & \text{else} \end{cases}$$

and, if the measurement variance is state-independent, $R(x) = R$. The ocean temperature can change stochastically by (5.10).

Notice that the subsystem (5.10), (5.11) satisfies the assumptions of linear-time varying Kalman Filters [70, Ch. 4], since for any given trajectory $x(t)$, the measurement model is equivalent to $y = C(t)m + v(t)$, where $C(t) = C(x(t))$ by slight abuse of notation. Therefore,

the estimate has distribution $\mathcal{N}(\mu, P)$, where μ, P evolve according to:

$$\dot{\mu} = PC(x)R(x)^{-1}(y - C(x)\mu), \quad \dot{P} = Q - \frac{C(x)^2}{R(x)}P^2.$$

Since the clarity of a scalar Gaussian is $q = 1/(1+P)$,

$$\dot{q} = \frac{\partial q}{\partial P}\dot{P} = \frac{-\dot{P}}{(1+P)^2} = \frac{-1}{(1+P)^2}\left(Q - \frac{C(x)^2}{R(x)}P^2\right)$$

and therefore the clarity dynamics are

$$\dot{q} = \frac{C(x)^2}{R(x)}(1-q)^2 - Qq^2. \tag{5.12}$$

Remark 5.2. *Comparing* (5.9) *with* (5.12)*, one may note that their structure is remarkably similar. Clarity/coverage increase due to the first term, and decrease due to the second. However,* (5.12) *is nonlinear wrt* q*. Thus, although* (5.9) *has the right intuitive characteristics to describe 'coverage',* (5.12) has the correct dynamics corresponding to information gathering.

Eq. (5.12) yields further insight. Clarity decays at a rate $-Qq^2$, i.e., due to the environment stochasticity. As clarity increases, the rate of increase of clarity, $C(x)^2(1-q)^2/R(x)$, decreases: additional measurements have diminishing value.

Although nonlinear, (5.12) has closed-form solutions, since it is a scalar differential Riccati equation [82, Sec. 2.15]. For constant $C(x) = C, R(x) = R$, if $C, R, Q > 0$,

$$q(t) = q_\infty\left(1 + \frac{2\gamma_1}{\gamma_2 + \gamma_3 e^{2kQt}}\right), \tag{5.13}$$

where $k = C/\sqrt{QR}$, $q_\infty = k/(k+1)$, $\gamma_1 = q_\infty - q_0$, $\gamma_2 = \gamma_1(k-1)$, $\gamma_3 = (k-1)q_0 - k$.

As $t \to \infty$, clarity monotonically approaches $q_\infty < 1$: if m is stochastic with non-zero variance, and measurements have non-zero variance, perfect clarity ($q = 1$) is impossible.

Theorem 5.3. *Let* $m \in \mathbb{R}^{n_m}$ *be the environment state vector, and* $y \in \mathbb{R}^q$ *be the sensed outputs. Suppose the environment and measurement models are*

$$\dot{m} = Am + w(t) \qquad w(t) \sim \mathcal{N}(0, Q) \tag{5.14a}$$

$$y = C(x)m + v(t) \qquad v(t) \sim \mathcal{N}(0, R(x)) \tag{5.14b}$$

with $Q \in \mathbb{S}^{n_m}_{++}$*, and* $R : \mathcal{X} \to \mathbb{S}^q_{++}$*. Assuming* $P(t) \in \mathbb{S}^{n_m}_{++}$ *for all* t *(see [91, Sec. 11.2]) and*

a prior $m \sim \mathcal{N}(\mu, P)$*, then*

$$\dot{P} = AP + PA^T + Q - PC(x)^T R(x)^{-1} C(x)P \tag{5.15}$$

$$\dot{q} = q(1-q)\left(\operatorname{tr}\left(C(x)^T R^{-1} C(x)P\right) - \operatorname{tr}\left(2A + P^{-1}Q\right)\right). \tag{5.16}$$

Proof. Eq. (5.15) is the standard covariance update for the Kalman Filter. To derive (5.16), notice the clarity of a multivariate Gaussian is $q = 1/(1+|P|)$. Therefore,

$$\dot{q} = -\frac{1}{(1+|P|)^2}\frac{d}{dt}(|P|)$$

Since $P \in \mathbb{S}^{n_m}_{++}$, it is invertible. Using Jacobi's formula:

$$\dot{q} = \frac{-|P|\operatorname{tr}\left(P^{-1}\dot{P}\right)}{(1+|P|)^2} = q(1-q)\operatorname{tr}\left(-P^{-1}\dot{P}\right)$$

since $|P|/(1+|P|)^2 = q(1-q)$. Substituting in (5.15), and simplifying, we arrive at (5.16). □

Again, we see the same structure: clarity increases at a rate $\operatorname{tr}\left(C(x)^T R(x)^{-1} C(x)P\right)$, and decreases at a rate $\operatorname{tr}\left(P^{-1}Q\right)$. Furthermore, since (5.15, 5.16) are independent of y, for trajectory planning we can use the deterministic and fully known

$$\dot{X} = \tilde{f}(X, u), \quad \dot{q} = g(X, q), \tag{5.17}$$

where $X = [x^T, \operatorname{vec}(P)^T]^T$ is an extended state.

5.1.2 Perceivability

In this section, we introduce the concept of *perceivability*: given a robot with certain sensing and actuation capabilities, can the robot's motion over a finite time achieve a desired level of clarity with the collected sensory data? Formally,

Definition 5.4. *A quantity* $m \in \mathbb{R}$ *that evolves according to* (5.10) *is* ***perceivable*** *by the system (5.8, 5.11) with clarity dynamics*[2] $g : \mathcal{X} \times [0,1] \to \mathbb{R}$*, to a level* $q^* \in [0,1]$ *at time* T *from an initial state* $x_0 \in \mathcal{X}$ *and clarity* $q_0 \in [0,1]$*, if there exists a controller* $\pi : [0,T] \to \mathcal{U}$ *s.t. the solution to*

$$\begin{bmatrix}\dot{x}\\ \dot{q}\end{bmatrix} = \begin{bmatrix} f(x, \pi(t)) \\ g(x,q)\end{bmatrix}, \quad \begin{bmatrix} x(0) \\ q(0)\end{bmatrix} = \begin{bmatrix} x_0 \\ q_0\end{bmatrix} \tag{5.18}$$

[2]When using a Kalman Filter to estimate m, g is as in (5.12). In general, other estimators could be used, and will lead to different expressions for g.

satisfies $q(T) \geq q^*$.

We define the set of initial conditions from which m is perceivable as the *perceivability domain*:

Definition 5.5. *The* (q^*, T)-***Perceivability Domain*** *of a quantity* $m \in \mathbb{R}$ *(that evolves according to* (5.10)*) by the system (5.8, 5.11) is the set of initial states* x_0 *and initial clarities* q_0 *such that* m *is perceivable to a level* q^* *at time* T*:*

$$\begin{aligned}\mathcal{D}(q^*, T) = \Big\{ (x_0, q_0) : \exists \pi : [0, T] \to \mathcal{U}, \\ \dot{x} = f(x, \pi(t)),\ \dot{q} = g(x, q), \\ x(0) = x_0,\ q(0) = q_0,\ q(T) \geq q^* \Big\}. \end{aligned} \tag{5.19}$$

Our key insight is that perceivability is fundamentally a question of the reachability of the augmented system (5.18). As with backward reachable sets, the perceivability domain can be determined by a Hamilton-Jacobi-Bellman (HJB) equation:

Theorem 5.4. *Let* $V : [0, T] \times \mathcal{X} \times [0, 1] \to \mathbb{R}$ *be the viscosity solution of*

$$\frac{\partial V}{\partial t} + \max_{u \in \mathcal{U}} \left(\frac{\partial V}{\partial x} f(x, u) \right) + \frac{\partial V}{\partial q} g(x, q) = 0, \tag{5.20a}$$

$$V(T, x, q) = q \quad \forall x \in \mathcal{X}, q \in [0, 1]. \tag{5.20b}$$

Then the (q^*, T)*- perceivability domain of* $m \in \mathbb{R}$ *(that evolves according to* (5.10)*) by the system (5.8, 5.11) is*

$$\mathcal{D}(q^*, T) = \left\{ [x_0^T, q]^T : V(0, x_0, q_0) \geq q^* \right\}. \tag{5.21}$$

Proof. Let $\mathcal{L}([t, T], \mathcal{U})$ be the set of piecewise continuous functions $\pi : [t, T] \to \mathcal{U}$. Define V as the maximum clarity reachable from (t, x, q):

$$V(t, x(t), q(t)) = \max_{\pi \in \mathcal{L}([t,T], \mathcal{U})} q(T) \text{ s.t. } (5.18)$$

By the principle of dynamic programming, for any $\delta > 0$,

$$V(t, x(t), q(t)) = \max_{\pi \in \mathcal{L}([t,t+\delta], \mathcal{U})} V(t + \delta, x(t + \delta), q(t + \delta))$$

Using a Taylor expansion about $\delta = 0$, as $\delta \to 0$,

$$V(t, x(t), q(t)) = \max_{u \in \mathcal{U}} \Big(V(t, x(t), q(t)) + \frac{\partial V}{\partial t}\delta + \frac{\partial V}{\partial x} f(x, u)\delta + \frac{\partial V}{\partial q} g(x, q)\delta \Big)$$

which simplifies to (5.20). □

After solving V, the optimal controller is [40, Ch. 4.2]

$$\pi(t, x, q) = \operatorname*{argmax}_{u \in \mathcal{U}} \left(\frac{\partial V}{\partial x} f(x, u) \right) \tag{5.22}$$

5.1.3 Simulations and Applications

Code and videos are available at https://github.com/dev10110/Clarity-and-Perceivability.

5.1.3.1 Energy-Aware Information Gathering

This example demonstrates the diminishing value of measurements. Consider the quadrotor tasked with measuring ocean temperature. It must fly to a target location, spend T seconds collecting information, and fly back. As T increases, more measurements are made and hence greater clarity is achieved, but at an energy cost. We wish to optimize T to maximize clarity and minimize energy. We model the energy cost as $E(t) = p_0 + p_1 T$, where p_0 is the energy cost of flying to and back from the target, p_1 is the hovering power draw.

The pareto front of $q(T)$ against $E(T)$ is depicted in Fig. 5.1. The diminishing value of measurements is clearly visible, as between $T \in [160, 320]$ s, the clarity only increases by 2.6%, but increases by 49.7% for $t \in [10, 20]$ s. To maximize the clarity/energy ratio, the quadrotor should collect measurements for $T^* = 57.4$ seconds (green tangent).

5.1.3.2 Coverage Control based on Clarity

Next, we demonstrate how clarity can be used in ergodic coverage controller of [115]. The robot is exploring a unit square, but certain regions have a greater target clarity than others, as labelled in Fig. 5.2a. The challenge with ergodic controllers is defining the fraction of time spent at each position p, and uniform allocation is often used as a heuristic. Since the target clarity has been specified, we can invert (5.13) to determine the appropriate time allocation.

Fig. 5.2 compares the behaviour of three coverage controllers: (A) a greedy controller hovers at the point p with maximum $(q_T(p) - q(t, p))$ until q_T is reached, (B) the ergodic

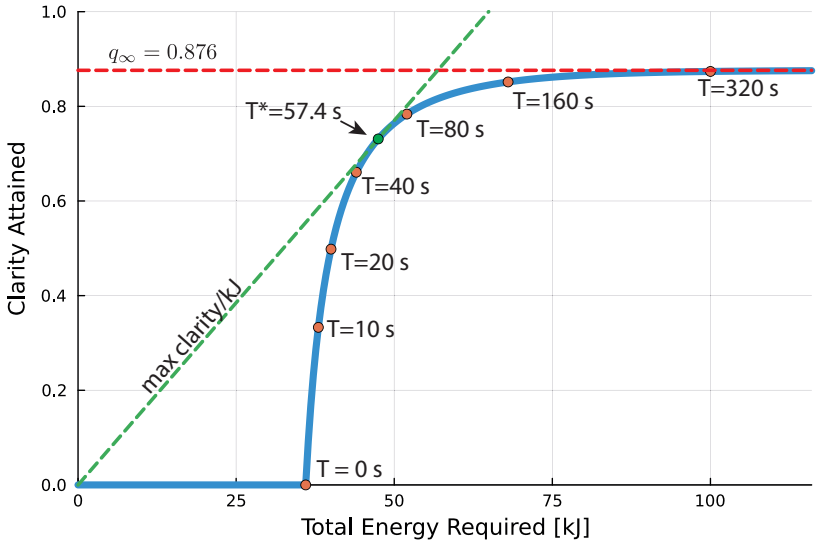

Figure 5.1: Clarity gained as a function of the measurement time. First, the clarity increases rapidly. As the level of clarity approaches q_∞ (red dashed line), the rate of clarity accumulation decreases. The maximum clarity/energy ratio is (green dashed line) is achieved at $T^* = 57.4$ s. Parameters: $R = 20.0$, $Q = 0.001$, $p_0 = 36$ kJ, $p_1 = 0.2$ kW.

controller in [115] with a uniform target distribution, and (C) the same ergodic controller but with a target distribution based on clarity. The proposed method (C) brings the mean of $(q(t,p) - q_T(p))$ to 0 rapidly, and does not overshoot like controller B. Beyond $t = 35$, $q(t,p)$ increases further since the robot continues to explore despite most cells having reached the target clarity.

5.1.3.3 Perceivability and Optimal Trajectory Generation

Here we demonstrate how perceivability can be determined using (5.20). Consider a boat tasked with collecting information that can only be measured from the green region in Fig. 5.3b. To highlight the importance of actuation capabilities on perceivability, we consider two models, a single integrator:

$$\dot{x}_1 = u_1 + w_x(x),\ \dot{x}_2 = u_2 + w_y(x)$$

with $u_1, u_2 \in [-2, 2]$ m/s, and a Dubins Boat:

$$\dot{x}_1 = v\cos x_3 + w_x(x),\ \dot{x}_2 = v\sin x_3 + w_y(x),\ \dot{x}_3 = u$$

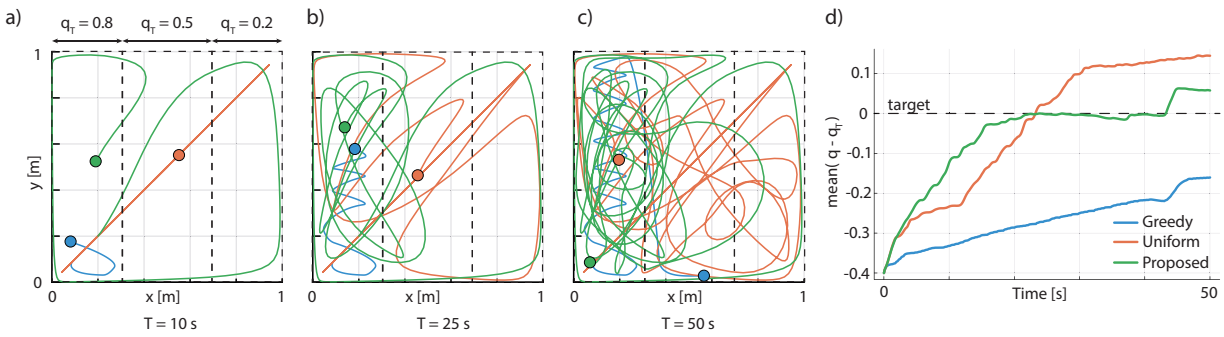

Figure 5.2: Coverage Controllers. (a-c) Snapshots of three controllers exploring a square region. The target clarity $q_T(p)$ is different in different regions as labelled in (a). (d) Plot of the mean$(q(t,p) - q_T(p))$ against t for each controller. Notice that using the proposed method, the mean clarity error is close to 0 for $t \in [20 - 35]$ seconds, and only increases later, when the entire region has higher clarity than the targets specified.

where $v = 2$ m/s, and $u \in [-1, 1]$ rad/s. For both, the sensing model is as in (5.12), with $C(x) = 1$ when x is in the green square and 0 elsewhere, $R(x) = 1.0$, $Q = 0.001$. The ocean current is $w_x(x) = \max(0, 3x_2), w_y(x) = -0.5$ m/s. Thus, neither vehicle has sufficient control authority to remain within the sensing range indefinitely.

To determine the perceivability domain, the backwards reachability set of (5.18) is computed using [25, 120] (Fig. 5.3a). The optimal controller (5.22) drives both vehicles from the same initial condition (Fig. 5.3b). Due to the current, both vehicles need to do loops to acquire clarity. The single integrator (Fig. 5.3c) is able to reach $q(T) \geq q^*$, while Dubins boat is not. Despite having the same sensing capabilities, the perceivability is different due to different actuation capabilities.

Computing the 10-second perceivability domain took 450 seconds on a Macboook Pro (i9, 2.3GHz, 16GB). While prohibitively slow for online applications, V can be precomputed offline. Future work will explore faster trajectory design techniques, akin to RIG [79], or CBFs, as demonstrated next.

5.1.3.4 CBF-based Trajectory Generation

Here we demonstrate how Lyapunov methods can be used to efficiently design controllers that maintain an information constraint, avoiding the need to numerically solve the HJB equation. Consider a 6D planar quadrotor system [5],

$$\ddot{x}_1 = u_1 \sin x_3/m,\ \ddot{x}_2 = u_1 \cos x_3/m - g,\ \ddot{x}_3 = u_2/J$$

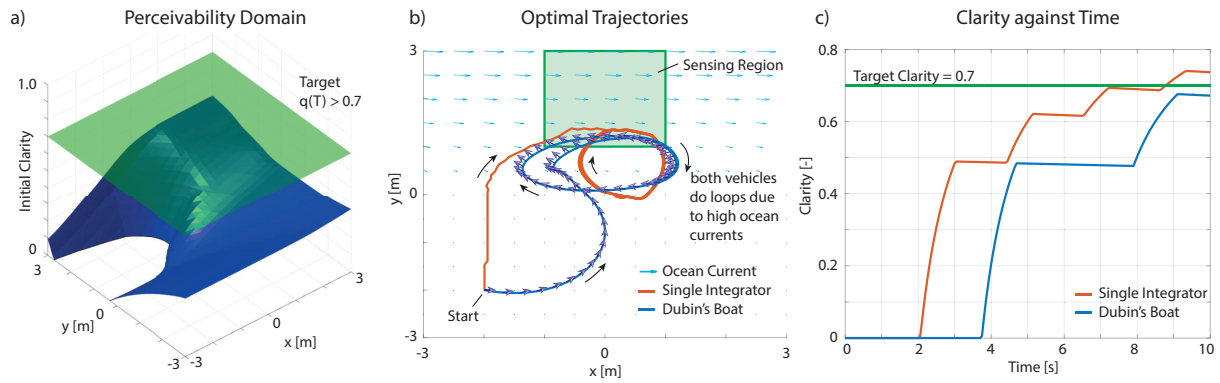

Figure 5.3: Perceivability and Optimal Trajectories. (a) The (q^*, T)-Perceivability Domain (states above blue surface) for single integrator using $q^* = 0.7, T = 10.0$ sec. (b) Optimal trajectories for the single integrator (orange) and the Dubins boat (blue) from the same initial conditions. The heading of Dubins boat is shown with blue arrows. Due to the high ocean currents in the sensing region, both vehicles make multiple passes through the sensing region to accumulate clarity. (c) Plot of clarity against time for both vehicles. Since the single integrator is more maneuverable than the Dubins boat, the environment is perceivable to a level 0.7 in time 10 seconds for the single integrator but not for the Dubin's boat.

where x_1, x_2 is the position of the quadrotor in the vertical plane, and x_3 is the pitch angle. m, g, J are the mass, acceleration due to gravity, and moment of inertia. The quadrotor is attempting a precision landing, using onboard sensors to determine the landing spot x_f. Given an estimate $\hat{x}_f$, an optimal control problem (OCP) can be solved to reach $\hat{x}_f$. However, since $\hat{x}_f$ is estimated online, we must ensure $\hat{x}_f$ is accurate before approaching it. This can be encoded as $\sigma \leq x_2/2$, which ensures that σ, the standard deviation of the estimated landing site is less than half the altitude, x_2. A constrained OCP can be defined, but is numerically difficult to solve since there are 7 state and 2 input dimensions.[3]

Instead, Lyapunov methods can be used to maintain the constraint. Using $\sigma^2 = 1/q - 1$, the safe set is

$$\mathcal{S} = \{[x^T, q]^T : h(x, q) = q - 4/(4 + x_2^2) \geq 0\}$$

where h is a CBF of relative degree 2 [21]. Fig. 5.4 compares the trajectories with and without the CBF-QP controller [176]. With the CBF-QP controller the quadrotor slows down to ensure high quality of information. Each iteration of the controller takes about 1 ms, significantly faster than HJB methods. This illustrates that by framing problems of information-based control using clarity/perceivability, Lyapunov methods can be used to design controllers to maximize (or in this case maintain) the quality of information gathered

[3]It took 480 s to compute the 0.05 s horizon value function on a coarse grid. Over a finer grid, the RAM usage exceeded 60GB and MATLAB crashed.

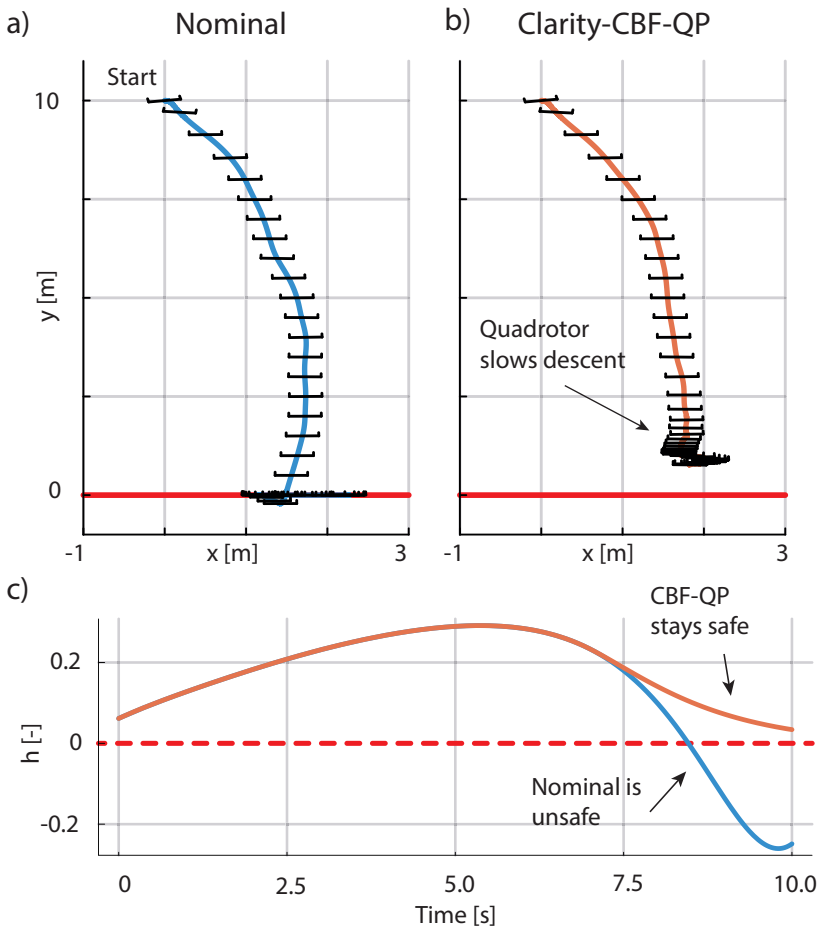

Figure 5.4: Precision landing of a planar quadrotor. (a) In the nominal controller, the quad descends rapidly and misses the target. (b) Using the clarity based CBF-QP controller, the quad descends slowly. (c) Plot of h against t, showing the CBF-QP keeps the system safe.

by a system.

5.1.4 Conclusion

The primary purpose of this section is to introduce perceivability, the ability of a robotic system to obtain information about the environment contingent on its actuation capabilities, its sensing capabilities, the environment's dynamics. As a metric for information, we introduce clarity, a bounded rescaling of differential entropy that takes values in $[0, 1]$. We have shown how perceivability is linked to a reachability problem of an augmented state, and through HJB equations, simultaneously determine a system's perceivability and the optimal control policy to maximize the final clarity. By using clarity, the HJB-based algorithms can be evaluated numerically, since (A) the range of the information state is bounded, and (B) the clarity dynamics have finite derivative. In the simulations, we demonstrate other ways

that the clarity and perceivability can be used, from minimizing the energy cost, or designing CBF-QP controllers to maintain a desired level of information.

Here, we considered stochastic environments and measurement models, but deterministic robot dynamics. The important case of stochastic robot dynamics will be studied in the future. We also intend to investigate a potential connection with data-informativity [166]: perhaps data-informative trajectories can be designed using perceivability, to improve data-driven system identification.

5.2 Controllers for Multiagent Information Gathering

A standard robotic mission is the collection of information that varies both in time and space over a domain of interest. To collect such information optimally, a (team of) robot(s) must reason about the currently available information, the target level of confidence in the information sought, the spatiotemporal evolution of the underlying information, and the robot's sensing capabilities, and (in the case of a team) coordinate the actions of each robot.

The design of informative path planners and dynamic coverage controllers has long been of interest [27, 106, 127], with a variety of techniques proposed including Voronoi partitioning [52], sampling approaches [39, 121], grid/graph based approaches [43, 175] and ergodic search [60, 115].

Here, we define the informative path planning or coverage control problem as follows: we have a team of robots that, at a fixed sampling frequency, measure the spatiotemporal environment at their respective positions. Using these measurements, we update our estimate of the state of the environment (referred to as information assimilation), while simultaneously controlling the robots position to determine the next location from which a measurement should be taken (referred to as the coverage controller). As such, the goal is to design a controller and information assimilation algorithm that efficiently reduce the uncertainty of the estimate of the state of the environment.

We quantify the uncertainty of a stochastic variable using an information-theoretic metric *clarity*, introduced in [6]. In particular, as the uncertainty of the stochastic variable decreases, (i.e., its differential entropy approaches $-\infty$), the clarity of the random variable approaches 1. Similarly, as the uncertainty increases, the clarity approaches zero.

We model the environment as a spatiotemporal field $f(t, p)$, i.e., a scalar function that varies in time and space; as an example, if the goal is to estimate the windspeed over a spatial and temporal domain, $f(t, p)$ represents the windspeed at any given time t and position p. The estimate is a function $\hat{f}(t, p)$ for each t, p, with an associated clarity $q(t, p)$ at each t, p. Numerically the state of the environment is a vector representing $\hat{f}(t, p)$ at a set of grid points. By taking (noisy) measurements of f using the robots at their respective locations, we can improve our estimate $\hat{f}$ and increase its clarity (i.e., reduce the uncertainty). At the same time, due to the time-varying nature of f, the clarity of $\hat{f}$ decreases for all points not being measured. This balance of information gain and decay will be an important element in designing the algorithms.

A key limitation of many of the methods listed above is that simplified heuristics are used to motivate the cost functions used in the informative path planners. For example, the ergodic search approaches assume that a Target Spatial Distribution (TSD) (defined as

the desired percentage of time that the robot should spend at any position in the domain) is provided by the user. However, there has been less work on how one can obtain such a target distribution in a principled manner taking into account the sensing capabilities of the robot or the temporal evolution of the state of the environment.

The goal of this section is to demonstrate how the cost function in informative path planning can be designed in a principled manner based on the assumed model of the environment. In particular, when estimating a spatiotemporal field, a common practice is to model it as a realization of a GP [171], and use the robot's measurements to update the estimate of the state of environment.

Here, we use the connection between GPs and Stochastic Differential Equations (SDEs) [44, 96, 145] to analyze the information-gathering capabilities of the robots: given the robot's take measurements at their respective locations, how much does the uncertainty in our estimate of state of the environment reduce? We answer this by quantifying a robot's sensing function and the environment's information decay function. For a point p in the domain, the sensing function defines the rate of increase of clarity at p due to measurements from a robot at position r. The decay function quantifies the rate of decrease of clarity due to the time-varying nature of $f(t, p)$. We use these functions to design coverage controllers that respect the rate of change of clarity when designing trajectories.

We make three main contributions: (A) We use clarity [6] to quantify the rate of change of uncertainty at a position p due to measurements made by a robot at a (possibly different) position r. Integrated over the mission domain, this quantifies the value of the robot being at position r. (B) We use this relation to propose two coverage controllers. (C) Being feedback controllers, we show how they scale naturally to the multi-agent setting. Finally, we demonstrate the algorithms using a realistic simulation, where a team of aerial robots explore a region of Austria, and estimate the wind speed over this region.

The two coverage algorithms proposed bear resemblance to the controllers in [27] and [126]. The first, referred to as the *direct controller*, directly chooses a control input to maximize the clarity of the state of the environment. The second, referred to as the *indirect controller*, computes a TSD based on the time required to increase the clarity to a given target value.

5.2.1 Preliminaries

We consider a problem with N_R robots, exploring a d-dimensional domain $\mathcal{D} \subset \mathbb{R}^d$. Each robot has a state in $\mathcal{X} \subset \mathbb{R}^n$, $n \geq d$. The state of the environment will be represented numerically at a set of N_G grid points.

5.2.1.1 Clarity

The information metric *clarity* was introduced in [6] and is based on differential entropy:

Definition 5.6. *[160, Ch. 8] The* ***differential entropy*** $h[X] \in (-\infty, \infty)$ *of a continuous random variable* X *with support* S *and density* $\rho : S \to \mathbb{R}$ *is*

$$h[X] = -\int_S \rho(x) \log \rho(x) dx. \tag{5.23}$$

Notice that as the uncertainty in X decreases, the entropy approaches $h[X] \to -\infty$. Clarity is defined in terms of differential entropy.

Definition 5.7. *Let* X *be a* n*-dimensional continuous random variable with differential entropy* $h[X]$*. The* ***clarity*** $q[X] \in (0, 1)$ *of* X *is defined as:*

$$q[X] = \left(1 + \frac{\exp\left(2h[X]\right)}{(2\pi e)^n}\right)^{-1}. \tag{5.24}$$

In other words, the clarity $q[X]$ about a random variable X lies in $(0, 1)$, where $q \to 0$ corresponds to the case where the uncertainty in X is infinite, and if X is perfectly known in an idealized (noise-free) setting, $q[X] = 1$. For a scalar Gaussian random variable $X \sim \mathcal{N}(\mu, \sigma^2)$, the clarity is $q[X] = 1/(1 + \sigma^2)$.

In an estimation context, we use clarity to quantify the quality of our estimate: as the clarity increases towards 1, the uncertainty of our estimate decreases towards 0. In [6] it was shown that when X is estimated using a Kalman filter, the clarity dynamics of the estimate of X can be obtained in closed form.

5.2.1.2 Gaussian Processes

A GP [171, Ch. 2] is a (scalar) stochastic process that is fully defined by the mean function $\mathsf{m} : \mathcal{D} \to \mathbb{R}$ and a kernel $\mathsf{k} : \mathcal{D} \times \mathcal{D} \to \mathbb{R}$:

$$f(p) \sim \mathsf{GP}(\mathsf{m}(p), \mathsf{k}(p, p')), \tag{5.25}$$

where m and k are defined as

$$\mathsf{m}(p) = E[f(p)], \tag{5.26a}$$

$$\mathsf{k}(p, p') = E[(f(p) - \mathsf{m}(p))(f(p') - \mathsf{m}(p'))]. \tag{5.26b}$$

Given a set of N measurements $\{y_k\}_{k=1}^N$ taken at positions $\{p_k\}_{k=1}^N$, we can update our posterior estimate of f, as described in [171, Ch. 2].

For two set of points $P_A = \{a_i\}_{i=1}^N$ and $P_B = \{b_i\}_{i=1}^M$, the kernel matrix $\mathsf{K}_{AB} \in \mathbb{R}^{N\times M}$ is the matrix such that $[\mathsf{K}_{AB}]_{(i,j)} = \mathsf{k}(a_i, b_j)$.

5.2.1.3 Spatiotemporal Gaussian Processes

The goal is to estimate a spatiotemporal field, i.e., to estimate a function $f(t,p)$, $f : \mathbb{R}\times\mathcal{D} \to \mathbb{R}$ using measurements obtained by robots.[4] Here $t \in \mathbb{R}$ denotes time, and $\mathcal{D} \subset \mathbb{R}^d$ is spatial domain of interest. The measurements (defined in (5.35)) are noisy measurements of f at a fixed sampling period from each robot's position at the sampling time.

While a standard GP can directly handle the spatiotemporal case, we can achieve significant computational efficiency by explicitly separating the spatial and temporal dimensions and exploiting the equivalence between spatiotemporal GPs and SDEs. Effectively, we can convert a Bayesian inference problem into a Kalman Filtering problem, thereby reducing memory and computational cost. We assume the following:

Assumption 5.1. *Suppose the spatiotemporal field $f : \mathbb{R}\times\mathcal{D} \to \mathbb{R}$ is a realization of a zero-mean GP:*

$$f(t,p) \sim \mathsf{GP}(0, \mathsf{k}(t,p,t',p')), \tag{5.27}$$

$$\mathsf{k}(t,p,t',p') = \mathsf{k}_\mathsf{T}(t,t')\mathsf{k}_\mathsf{S}(p,p'), \tag{5.28}$$

where the kernel is separable in space and time, and the temporal kernel is isotropic, i.e., $\mathsf{k}_\mathsf{T}(t,t')$ only depends on $|t'-t|$.

Under Assumption 5.1, it is known that realizations of a GP are also realizations of a SDE [44]. This fact is derived through the Wiener-Khinchin theorem [146, Ch. 12], and in the interest of space, the readers are referred to [44] or [7, Appendix] for full derivations.

The key idea is that if $h(t) \sim \mathsf{GP}(0, \mathsf{k}_\mathsf{T}(t,t'))$ is a realization of a (temporal) GP, it is equal to the output of a transfer function applied to a realization of a white noise process. By expressing the transfer function in state-space form, we arrive at a SDE such that a realization of the SDE is equal to h.

In the spatiotemporal case, let $P_G = \{p_i\}_{i=1}^{N_G} \subset \mathcal{D}$ be a set of N_G (possibly non-uniform) grid points over the spatial domain. Let $\boldsymbol{f}(t) \in \mathbb{R}^{N_G}$ be a vector such that the i-th entry is the value of the spatiotemporal field at the i-th grid point, $[\boldsymbol{f}(t)]_i = f(t,p_i)$. Then, the SDE

[4] For simplicity of exposition, we assume the spatiotemporal field has scalar outputs. For multidimensional outputs, we repeat for each dimension independently.

for the system comprises of N_G independent stochastic processes (5.29a), that get spatially correlated based on the spatial kernel (5.29b). Mathematically,

$$\begin{cases} ds_i(t) = As_i(t)dt + BdW_i(t) \\ z_i(t) = Cs_i(t) \\ s_i(0) \sim \mathcal{N}(0, \Sigma) \end{cases}, \tag{5.29a}$$

$$\boldsymbol{f}(t) = \sqrt{\mathsf{K_{GG}}}\boldsymbol{z}(t) = \sqrt{\mathsf{K_{GG}}}(I_{N_G} \otimes C)\boldsymbol{s}, \tag{5.29b}$$

Here $s_i(t) \in \mathbb{R}^{n_k}$ is a state at each grid point.[5] $\boldsymbol{s} = \begin{bmatrix} s_1^T & \cdots & s_G^T \end{bmatrix}^T \in \mathbb{R}^{n_k N_G}$ is a stacked vector representing the state of the entire environment; $\boldsymbol{f}(t) = \begin{bmatrix} f(t, p_1) & \cdots & f(t, p_G) \end{bmatrix}^T \in \mathbb{R}^{N_G}$ is a stacked vector comprising the value of the field at each grid point; W_i is a standard Wiener process, independent for each grid point. The matrices $A \in \mathbb{R}^{n_k \times n_k}$, $B \in \mathbb{R}^{n_k \times 1}$, $C \in \mathbb{R}^{n_k \times l}$ are constant matrices that only depend on the temporal kernel $\mathsf{k_T}$. $\Sigma \in \mathbb{S}_{++}^{n_k}$ is the matrix that solves $A\Sigma + \Sigma A^T = -BB^T$. $\mathsf{K_{GG}} \in \mathbb{S}_{++}^{\mathsf{G}}$ is the spatial kernel matrix, i.e., $[\mathsf{K_{GG}}]_{\mathsf{ij}} = \mathsf{k_S}(\mathsf{p_i}, \mathsf{p_j})$.

Example 5.3. The Matern-1/2 temporal kernel is $\mathsf{k_T}(t, t') = \sigma_t^2 \exp(-\lambda_t |t - t'|)$ for hyperparameters $\lambda_t, \sigma_t > 0$. The state-space model has dimension $n_k = 1$, and matrices $A = \begin{bmatrix} -\lambda_t \end{bmatrix}$, $B = \begin{bmatrix} 1 \end{bmatrix}$, $C = \begin{bmatrix} \sqrt{2\lambda_t}\sigma_t \end{bmatrix}$. Derivations and expressions for Matern-3/2 and Matern-5/2 kernels can be found in [7, Appendix].

5.2.1.4 Ergodic Control

Ergodic control [60, 115] is a technique to generate robot trajectories that cover a domain $\mathcal{D} = [0, L_1] \times \cdots \times [0, L_d] \subset \mathbb{R}^d$, such that the trajectories have a spatial (position) distribution that closely matches a specified TSD, as explained below.

The TSD is a function $\phi : \mathcal{D} \to \mathbb{R}$ such that the $\phi(p)$ denotes the desired time a robot should spend at position p. Given a robot's (position) trajectory $\xi : [0, T] \to \mathcal{D}$ the trajectory's spatial distribution is defined as $c_\xi : \mathcal{D} \to \mathbb{R}$, where for any $p \in \mathcal{D}$

$$c_\xi(p) = \frac{1}{T} \int_0^T \delta(p - \xi(\tau)))d\tau. \tag{5.30}$$

Here $\delta : \mathbb{R}^d \to \mathbb{R}$ is the Dirac delta function.

The *ergodicity* $E > 0$ of a trajectory ξ measures the difference between the robot trajec-

[5] n_k depends on the temporal kernel. For the Matern 1/2, 3/2, and 5/2 kernels, $n_k = 1, 2, 3$ respectively.

tory's spatial distribution and the target spatial distribution:

$$E = \|c_\xi - \phi\|^2_{H^{-s}} \tag{5.31}$$

where $\|\cdot\|_{H^{-s}}$ is the Sobolev space norm of order $s = (d+1)/2$, defined in [115]:

$$\|c_\xi - \phi\|^2_{H^{-s}} = \sum_{l \in \mathbb{N}^d} \Lambda_l (\hat{c}_l - \hat{\phi}_l)^2 \tag{5.32}$$

where $\Lambda_l \in \mathbb{R}$ is a weighting coefficient, and $\hat{(\cdot)}_l$ is the l-th element of the Discrete Cosine Transform (DCT) of the function $(\cdot)$, e.g.

$$\hat{\phi}_l = \langle \phi_l, b_l \rangle = \int_{p \in \mathcal{D}} \phi_l(p) b_l(p) dp \tag{5.33}$$

where $b_l : \mathcal{D} \to \mathbb{R}$ is the l-th basis function. We refer the reader to [115] for further details.

E is a function-space norm measuring the difference between the TSD and the spatial distribution of the trajectory. The key benefit of the Sobolev norm is that it prioritizes matching the low spatial frequency differences between c and ϕ before matching the high spatial frequencies. This means that the controllers have a multiscale-spectral nature, where they prioritize covering the domain globally, before returning to the gaps and covering them [115].

In [115] a feedback controller is derived for single and double-integrator robot models that minimizes the ergodicity. Various extensions have been presented in, for example, [59, 60] to address other robot models and other goals.

5.2.2 Problem Statement

Consider a team of $N_R > 0$ robots, each with dynamics

$$\dot{x}_i = F(x_i) + G(x_i) u_i, \tag{5.34}$$

where $x_i \in \mathcal{X} \subset \mathbb{R}^n$ is the i-th robot's state, and $u_i \in \mathcal{U} \subset \mathbb{R}^m$ is its control input. The position of each robot is $r_i = \Phi(x_i) \in \mathbb{R}^d$, i.e., $\Phi : \mathcal{X} \to \mathcal{D}$ extracts the position.

Each robot makes measurements of the spatiotemporal field at a fixed sampling period $\Delta T > 0$,

$$y_{k,i} = f(t_k, \Phi(x_i(t_k))) + w_{k,i}, \tag{5.35a}$$

$$w_{k,i} \sim \mathcal{N}(0, \sigma_m^2), \tag{5.35b}$$

that is, $y_{k,i} \in \mathbb{R}$ is a scalar measurement output by the i-th robot at the k-th timestep, $t_k = k\Delta T$. Each measurement is perturbed by zero-mean Gaussian noise with standard deviation σ_m.

We assume each robot determines its control inputs, but that the information from each robot is assimilated centrally. We assume the robots can always communicate with the central agent, sending the measurements and receiving a map of the current clarity at each $p \in \mathcal{D}$.

Problem 5.1. *Consider a team of $N_R > 0$ robots, each with dynamics* (5.34) *and measurements* (5.35)*, exploring a domain $\mathcal{D}$. Let $f : \mathbb{R} \times \mathcal{D} \to \mathbb{R}$ be a spatiotemporal field to be estimated satisfying Assumption 5.1. Design a coverage control algorithm for each robot, and an estimation algorithm to fuse measurements y_k into an estimate of f.*

The mathematical form of the coverage objective is delayed until Section 5.2.4. The estimator will be the optimal estimator in a least-squares sense, discussed in Section 5.2.3.

In addressing Problem 5.1, we address two questions: (A) how does the information assimilation algorithm inform the value of taking measurements at a robot position $x \in \mathcal{D}$ on the quality of information at a different position $p \in \mathcal{D}$, and (B) how should one design coverage controllers to exploit that relationship? Since the mission is a multi-agent coverage problem, we also need to ensure that the proposed coverage algorithms are scalable with the number of robots. We address these two questions in the following sections.

5.2.3 Information Assimilation

In this section, we discuss how the GP model (Assumption 5.1) determines two functions: (A) the information decay rate at each $p \in \mathcal{D}$, and (B) the information gain rate at each $p \in \mathcal{D}$ due to measurements taken from a robot's position $r_i = \Phi(x_i) \in \mathcal{D}$. We consider the hyperparameters of the GP to be specified and constant, although some strategies for estimating these are discussed in the simulation section.

5.2.3.1 Kalman Filter Model

First, we show that the Kalman Filter (KF) is the optimal state estimator to estimate the spatiotemporal field f. As shown in Section 5.2.1.3, the process model for f sampled at N_G grid points is a linear stochastic differential equation with state $\boldsymbol{s} \in \mathbb{R}^{n_k N_G}$. We now show that the measurements (5.35) are a linear function of $\boldsymbol{s}$.[6]

[6] In [44], the measurements must be taken at one of the grid points. Here we extend the result to allow measurements at non-grid points.

Consider N_R robots at positions $P_R = \{\Phi(x_i)\}_{i=1}^{N_r}$ where each robot makes a measurements $y_{k,i}$ as in (5.35). However, the state $\boldsymbol{s}$ corresponds to the grid points P_G, not necessarily coinciding with the measurement locations P_R. To account for this, we use spatial correlation based on the Gaussian Process model for f:

$$\begin{bmatrix} \boldsymbol{f}(t_k) \\ \boldsymbol{y}_k \end{bmatrix} \sim \mathcal{N}\left(0, \begin{bmatrix} \mathsf{K}_{GG} & \mathsf{K}_{GR} \\ \mathsf{K}_{RG} & \mathsf{K}_{RR} + \sigma_m^2 I \end{bmatrix}\right) \tag{5.36}$$

where $\mathsf{K}_{GG}, \mathsf{K}_{GR}, \mathsf{K}_{RG}, \mathsf{K}_{RR}$ are the kernel matrices for the sets of points P_G, P_R, and $\boldsymbol{y}_k = \begin{bmatrix} y_{k,1} & \cdots & y_{k,N_R} \end{bmatrix}^T$.

Using (5.29b), $\boldsymbol{y}_k$ conditioned on the state $\boldsymbol{s}(t_k)$ is

$$\boldsymbol{y}_k | \boldsymbol{s}(t_k) \sim \mathcal{N}(H\boldsymbol{s}(t_k), V), \tag{5.37a}$$

$$H = \mathsf{K}_{RG}\mathsf{K}_{GG}^{-1}\sqrt{\mathsf{K}_{GG}}(I_{N_G} \otimes C) \tag{5.37b}$$

$$V = \sigma_m^2 I_{N_R} + \mathsf{K}_{RR} - \mathsf{K}_{RG}\mathsf{K}_{GG}^{-1}\mathsf{K}_{GR}. \tag{5.37c}$$

Therefore, the environment's state space model is a linear (continuous time) process (recall (5.29a)) with linear (discrete-time) measurements:

$$d\boldsymbol{s} = (I_{N_G} \otimes A)\boldsymbol{s}dt + (I_{N_G} \otimes B)dW, \tag{5.38a}$$

$$\boldsymbol{y}_k = H\boldsymbol{s}(t_k) + v_k \tag{5.38b}$$

where W is a N_G-dimensional standard Wiener process, and $v_k \sim \mathcal{N}(0, V)$. Notice that although each measurement has noise variance $\sigma_m^2 I$, the noise model in (5.38b) has $V \geq \sigma_m^2 I$ accounting for the fact that measurements can be taken at non-grid points.

To summarize, we have a linear, time-invariance process model (5.38a), with a linear (but time-varying due to the changing measurement locations) measurement model (5.38b). Together, they satisfy the assumptions of the KF, and therefore the KF is the optimal estimator for this system [70].

5.2.3.2 Quantifying Information Gain and Decay Rates

Next, we wish to characterize the clarity dynamics, i.e., the rate of information gain and decay. In this section, we focus on the clarity dynamics of a single point $p \in \mathcal{D}$ due to a measurement taken by a robot with position $r = \Phi(x) \in \mathcal{D}$. Since we use the KF to assimilate measurements, we use the earlier derived dynamics to estimate the rate of information gain.

Reducing (5.38) for a single point p, the continuous time KF model is

$$\dot{s} = As + Bw, \qquad w(t) \sim \mathcal{N}(0, I), \tag{5.39a}$$

$$y = Ls + v, \qquad v(t) \sim \mathcal{N}(0, V\Delta T) \tag{5.39b}$$

where $s \in \mathbb{R}^{n_k}$ is the state of the spatiotemporal process at p, $r = \Phi(x)$ is the robot's position, and

$$L = \frac{\mathsf{k_S}(r,p)}{\sqrt{\mathsf{k_S}(p,p)}}C, \quad V = \sigma_m^2 + \mathsf{k_S}(r,r) - \frac{\mathsf{k_S}(r,p)^2}{\mathsf{k_S}(p,p)}.$$

Let the KF state consist of $(\hat{s}, \Sigma)$, the mean and covariance. Then, the covariance has dynamics

$$\dot{\Sigma} = A\Sigma + \Sigma A^T + BB^T - \Sigma L^T (V\Delta t)^{-1} L\Sigma. \tag{5.40}$$

Therefore, the estimate of $f(t,p)$ is $\mathcal{N}(\hat{f}, \Pi)$, where $\hat{f} = C\hat{s}$, and $\Pi = C\Sigma C^T$. Since, the clarity of a scalar Gaussian variable is $q = 1/(1+\Pi)$, the clarity dynamics are

$$\dot{q} = \frac{dq}{d\Pi}\dot{\Pi} = -q^2 C\dot{\Sigma}C^T. \tag{5.41}$$

Depending on the temporal kernel,[7] this simplifies to

$$\dot{q} = \underbrace{S(x,p)(1-q)^2}_{\text{clarity gain}} - \underbrace{D(p,q)}_{\text{clarity decay}} \tag{5.42}$$

where the first term defines the rate of clarity gain at p due to measurements taken at $r = \Phi(x)$, while the second term defines the clarity decay rate.

Remark 5.3. *Eq.* (5.42) *is one of our main results: the function* $S : \mathcal{X} \times \mathcal{D} \to \mathbb{R}$ *is the sensing function that quantifies the importance of a measurement taken from robot state* $x \in \mathcal{X}$ *on the clarity of our estimate at a position* $p \in \mathcal{D}$*. Similarly,* $D : \mathcal{D} \times \mathbb{R} \to \mathbb{R}$ *defines the rate at which clarity about* $f(t,p)$ *decays due to the spatio*temporal *nature of* f*. Notice the decay rate is uncontrolled, i.e., does not depend on the robot's state* x*.*

[7] In particular, this holds for Matern-1/2 kernels.

Example 5.4. For Matern-1/2 temporal kernels,

$$S(x,p) = \frac{1}{\Delta T}\frac{\mathsf{k_S}(r,p)^2}{\mathsf{k_S}(p,p)\left(\mathsf{k_S}(r,r)+\sigma_m^2\right) - \mathsf{k_S}(r,p)^2}$$
$$W(p,q) = 2\lambda_t\left((\sigma_t^2+1)q^2 - q\right),$$

where $r = \Phi(x)$ is the position of a robot at state x. Since for isotropic spatial kernels $\mathsf{k_S}(p,p') = \mathsf{k_S}(\|p-p'\|)$,

$$S(d) \propto \frac{\mathsf{k_S}(d)^2}{\mathsf{k_S}(0)^2 + \sigma_m^2\mathsf{k_S}(0) - \mathsf{k_S}(d)^2}$$

where $d = \|\Phi(x) - p\|$ is the distance at which the measurement is taken. When $d \mapsto \mathsf{k_S}(d)$ is nonincreasing, e.g. in the Matern and Squared Exponential kernels, $S(x,p)$ is maximized at $\Phi(x) = p$, implying that the rate of increase in clarity about p is maximized when the robot is also at position p. This is not, in general, true, since for example in periodic or polynomial spatial kernels, $S(x,p)$ may be maximized for some $\Phi(x) \neq p$. Furthermore notice that in the limiting case of a spatiostatic environment, $\lambda_t \to 0$, and therefore the decay rate $D(p,q) \to 0$.

To summarize, in this information-gathering problem the spatiotemporal information to be collected is modeled using a GP. To define a suitable coverage algorithm, we need to quantify the value of taking a measurement at some robot state $x \in \mathcal{X}$ on the clarity gain at any other position $p \in \mathcal{D}$. This is captured by the clarity dynamics (5.42). The key functions are S, the sensing function, and W, the decay function. Notice only $S(x,p)$ is controllable since it is the only term in (5.42) that depends on the robot's state x.

5.2.4 Coverage Controllers

In this section, we use the sensitivity and decay functions in (5.42) to derive two coverage controllers. The direct method chooses a control input that maximizes the rate of increase in the total clarity integrated over the domain $\mathcal{D}$. The indirect method determines the time that the robot should spend at each position in the domain to achieve a target clarity and then uses ergodic control to compute the control input.

5.2.4.1 Direct Method

The direct method minimizes the cost function

$$J(t) = \|\overline{q}(\cdot) - q(t,\cdot)\|_H^2, \tag{5.43}$$

a function-space norm over $p \in \mathcal{D}$ between the current clarity distribution $q(t,p)$ and the target clarity distribution $\overline{q}(p)$. We use the Sobolev norm, defined in (5.32), and discussed below.

Notice that $J(t)$ does not explicitly depend on the robot's state or control input. As such, we choose to minimize J over a short horizon $\delta > 0$ in the future:

$$J(t+\delta) \approx J(t) + \dot{J}(t,x)\delta^2 + \frac{1}{2}\ddot{J}(t,x,u)\delta^2 + \cdots \tag{5.44}$$

where the dependency on u first shows up in the $\ddot{J}$ term. This high-relative degree behavior is a consequence of the fact that the clarity dynamics (5.42) depend on x, not $\dot{x}$. Therefore the second derivative of J must be taken for the control input to appear in the expressions. This behavior is commonly observed in the literature on coverage control, as in [27, Ch. 2] and in [115]. Then, given control inputs $u \in \mathcal{U} \subset \mathbb{R}^m$, the controller will be of the form

$$\pi(t,x) = \underset{u \in \mathcal{U}}{\operatorname{argmin}}\ \ddot{J}(t,x,u). \tag{5.45}$$

We will derive a closed-form solution for this controller. Before doing so, we justify our choices for the cost function and the control strategy.

We use the Sobolev norm for the following reasons. In [27], a differentiable sensing functional (an analog of S) is used with the generalized transport theorem to compute an analog of $\ddot{J}(t,x,u)$. However, this approach often leads to local minima, where $\ddot{J}(t,x,u)$ becomes independent of u. This happens when all of the local information has been collected, and there is no preference for the controller to move in one direction over the other. To address this, [27] proposed combining the local search strategy with a global strategy, where the controller would choose a new global waypoint when the local controller reaches a local minimum. In our work, we use the Sobolev space norm instead of the ℓ_2 norm, and this allows the controllers to have a multispectral property [115] - it prioritizes global coverage before prioritizing local coverage.

Second, to evaluate $\ddot{J}$, we use the clarity dynamics we derived in Section 5.2.3.2. This is in contrast to earlier works that used heuristic expressions to quantify coverage, and coverage dynamics [27, 77]. As such, the derived controllers depend explicitly on the spatiotemporal field's kernel, and the sensing capabilities (in particular the sampling period ΔT and measurement noise σ_m) of the robots.

Next, we derive the controller. The cost function is

$$J(t) = \|\overline{q}(\cdot) - q(t,\cdot)\|_H^2 = \sum_{l \in \mathbb{N}^d} \Lambda_l \left(\hat{\overline{q}}_l - \hat{q}_l(t)\right)^2, \tag{5.46}$$

where $\hat{\overline{q}}_l = \langle \overline{q}, b_l \rangle$, $\hat{q}_l(t) = \langle q_l(t,\cdot), b_l \rangle$ are the inner products of $\overline{q}(\cdot)$ and $q(t,\cdot)$ with the l-th basis function of the DCT. Recall the notation $\langle a, b_l \rangle$, and Λ_l was defined in Section 5.2.1.4. After some algebraic calculations, one can show that the first and second time-derivatives of J are:

$$\begin{aligned}
\dot{J}(t,x) &= \sum_{l \in \mathbb{N}^d} -2\Lambda_l (\hat{\overline{q}}_l - \hat{q}_l(t)) \dot{\hat{q}}_l(t,x) \\
\ddot{J}(t,x,u) &= \sum_{l \in \mathbb{N}^d} 2\Lambda_l \left(\dot{\hat{q}}_l^2(t,x) - (\hat{\overline{q}}_l - \hat{q}_l(t)) \ddot{\hat{q}}_l(t,x,u) \right)
\end{aligned}$$

where $\dot{\hat{q}}_l(t,x), \ddot{\hat{q}}_l(t,x,u)$ are

$$\begin{aligned}
\dot{\hat{q}}_l &= \frac{d}{dt} \langle q(t,\cdot), b_l \rangle \\
&= \int_{p \in \mathcal{D}} \left(S(x,p)(1 - q(t,p))^2 - W(p, q(t,p)) \right) b_l(p) dp
\end{aligned}$$

where S is as defined in (5.42). Similarly,

$$\ddot{\hat{q}}_l = \frac{d^2}{dt^2} \int_{p \in \mathcal{D}} q(t,p) b_l(p) dp = \hat{B}_l(t,x)\dot{x} + \mathcal{O},$$

where $\mathcal{O}$ collects terms independent of $\dot{x}$ (and therefore u), and $\hat{B}_l(t,x) \in \mathbb{R}^{1 \times n}$ is as defined as

$$\hat{B}_l(t,x) = \left\langle (1 - q(t,\cdot))^2 \frac{\partial S}{\partial x}(x,\cdot), b_l \right\rangle. \tag{5.47}$$

Therefore, we have

$$\begin{aligned}
\ddot{J}(t,x,u) &= \sum_{l \in \mathbb{N}^d} -\Lambda_l (\hat{\overline{q}}_l - \hat{q}_l(t)) \hat{B}_l(t,x)\dot{x} + \mathcal{O} \\
&= -L(t,x)(F(x) + G(x)u) + \mathcal{O}
\end{aligned}$$

where we define

$$L(t,x) = \sum_{l \in \mathbb{N}^d} \Lambda_l (\hat{\overline{q}}_l - \hat{q}_l(t)) \hat{B}_l(t,x).$$

Therefore, the choice of u that minimizes $J(t+\delta)$ yields a feedback controller $\pi_{dir} : \mathbb{R} \times \mathcal{D} \to$

$\mathcal{U}$,

$$\pi_{dir}(t,x) = \underset{u\in\mathcal{U}}{\operatorname{argmin}} \; -L(t,x)G(x)u$$

If $\mathcal{U} = \{u \in \mathbb{R}^m : \|u\| \leq u_{max}\}$, and $L(t,x)G(x) \neq 0$,

$$\pi_{dir}(t,x) = u_{max}\frac{G(x)^T L^T(t,x)}{\|L(t,x)G(x)\|}. \tag{5.48}$$

Proving that $L(t,x)G(x) \neq 0$ for any t, x is non-trivial, and will be studied in future work.

5.2.4.2 Indirect Method

The second approach is inspired by ergodic control. Ergodic control uses a TSD to determine the feedback control law, as discussed in Section 5.2.1.4. Here we derive a principled method to construct the TSD based on the information assimilation algorithm discussed in Section 5.2.3.

The key idea is to set the TSD to be the time required for the clarity of our estimate of f to increase from its current value to a specified target clarity $\overline{q}(p)$, assuming the robot was making measurements from $x = p$. To compute this, we solve the differential equation (5.42) and determine $T(q, \overline{q})$, i.e., the time required to increase the clarity from q to $\overline{q}$.

Then, given the target clarity distribution $\overline{q} : \mathcal{D} \to [0,1]$, and the current clarity distribution $q(t,\cdot) : \mathcal{D} \to [0,1]$, the TSD can be specified as follows:

$$\text{TSD}(t,p) = \begin{cases} T(q(t,p), \overline{q}(p)) & \text{if } q(t,p) \leq \overline{q}(p) \\ 0 & \text{else} \end{cases}. \tag{5.49}$$

This equation has an analytic solution, see [7, Appendix].

Finally, we can use the ergodic control method described in [115] to design a feedback controller for the system,

$$\pi_{ind}(t,x) = \pi_{ergo}(t,x,\text{TSD}) \tag{5.50}$$

5.2.4.3 Extension to Multi-Robot Coverage Control

Our proposed coverage controllers have been presented for the single-robot cases above. Here we discuss the extension and implementation of these methods in the multi-agent case, where multiple robots have to decide how to move to collect information. We assume that they can synchronize their information by sharing the clarity map, $q(t,p)\ \forall p \in \mathcal{D}$, over a centralized

setting, i.e., that they are connected over a complete graph so that each robot has access to a centrally stored clarity map. The extensions to distributed settings are left for future work.

Notice that both proposed controllers are feedback controllers, depending on the robot's position, and the clarity map $q(t, p)$. Therefore, the control input for each agent can be computed as $u_i = \pi(t, x_i, q)$, where x_i denotes the position of the i-th agent, and $\pi \in \{\pi_{dir}, \pi_{ind}\}$ can be either control strategy. In the indirect approach, we must also share the history of positions visited by the agents.

As the robots move using the coverage controllers, the robots make measurements of the spatiotemporal field from their respective positions. These measurements are assimilated into a single estimate of the spatiotemporal field using the KF model. The information assimilation is currently performed centrally, although future work will look into distributed methods of maintaining the estimate.

5.2.5 Simulations

In this section, we report the simulation results of an information-gathering mission. As a prototypical example, we consider the collection of wind data using a team of ten aerial robots. The robots perform a two-hour mission, and we aim to maximize the clarity of the wind field over the domain by the end of the mission. Our evaluation metric is both the accuracy of the reconstruction, as well as the average clarity over the mission domain.

The mission domain is a 12.7×6.3 km^2 region of southeastern Austria, located near 46.93° N, 15.90° E, chosen because of a high-quality ground-truth data set available from WegenerNet [148]. The dataset provides wind speeds over the domain at a resolution of 100 m and 30 minutes. The mission domain is particularly challenging due to its high weather and climate variability [148]. Over the domain considered, the maximum wind speed is 13 m/s.

Each robot is capable of measuring the local x- and y-wind speed every 5 seconds. Each measurement is perturbed by noise with $\sigma_m = 0.5$ m/s. The robots are modeled as single-integrators with a maximum speed of 15 m/s. We use the KF model with a spatial grid resolution of 320 and 160 m in the x- and y-directions to model the state of the environment.

The spatial and temporal hyperparameters were estimated using techniques from geostatistics [50, 167]. In particular, we constructed a variogram of the dataset and used a least-squares fit to both the Matern-1/2 and the Squared Exponential kernels. The Matern-1/2 kernel fits the data better and is depicted in Figure 5.5b. The resulting kernel is of the form $k(t, p, t', p') = \sigma^2 \exp\left(-|t - t'|/l_t\right) \exp\left(-\|x - x'\|/l_s\right)$, where $\sigma = 2.11$ m/s, $l_t = 183$ min, $l_s = 1.61$ km. Fitting the kernel using the variogram was computationally much faster and

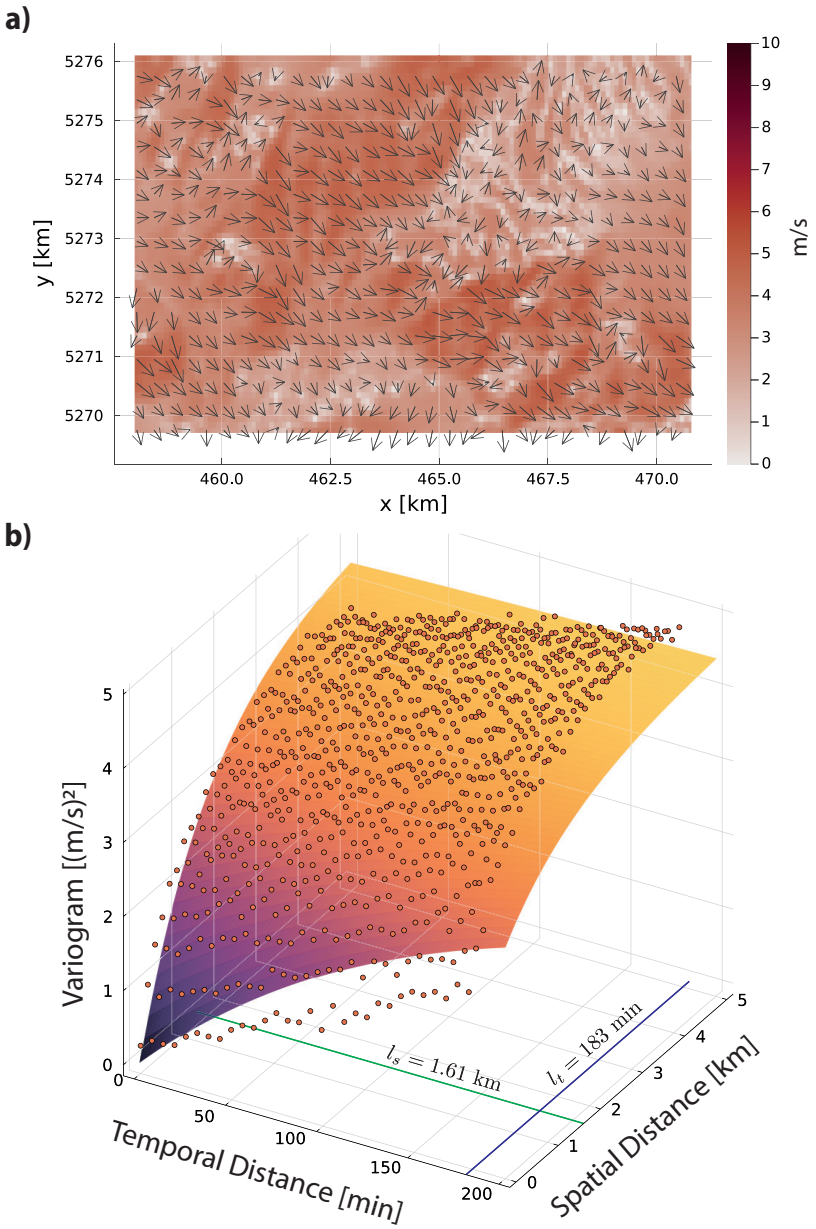

Figure 5.5: Wind data from WegenerNet [148]. (a) Wind speed and direction on Jan 1, 2023, 00:00, (b) Variogram showing the spatiotemporal correlation of the data. Surface shows the fitted kernel.

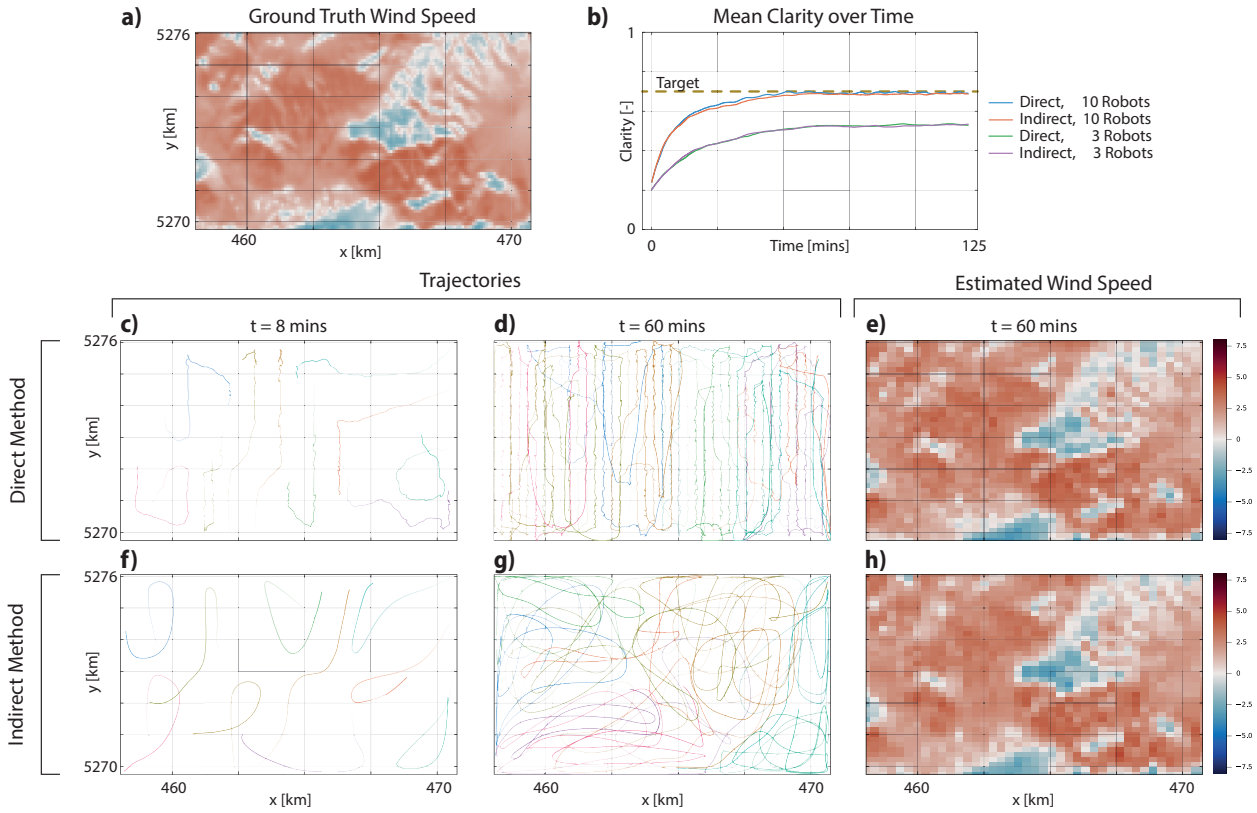

Figure 5.6: Simulation results. (a) shows the ground truth wind speed at the end of the simulation. (b) shows the mean clarity against time. The mean is taken spatially. (c-e) show the behavior of the direct method. (f-h) show the behavior of the indirect method. (c, d, f, g) show the trajectories of the ten robots after eight minutes and after sixty minutes. (e, h) show the estimated wind speed, and it closely matches the ground truth in (a).

more reliable than the nonlinear minimization of the log-likelihood method of [171]. See [7, Appendix] for additional details.

Simulations were run using both the direct and the indirect control strategies, and the results are summarized in Figure 5.6. Fig. 5.6(a) shows the ground-truth data to be estimated.[8]

Fig. 5.6(b) shows the change in average clarity over time as the robots explore the environment. Both the direct and indirect methods result in an almost identical average clarity at each timestep. Furthermore, after about an hour of exploration, the average clarity reaches a steady state value. This shows that due to the information decay rate, even as the robots continually explore the environment, the average clarity cannot be increased further.

Fig. 5.6(c,d,f,g) show the trajectories using both controllers. Figs. 5.6(c, d) show the trajectories of the direct method after 8 and 60 mins, and Figs. 5.6(f, g) show the corresponding trajectories of the indirect method. The trajectories generated by the two methods are remarkably different - in the direct method, the trajectories are jagged and tend to follow straight lines. This is because of $\partial S/\partial x$ in (5.47), which places significant benefit on local data collection. In contrast, the indirect method creates smoother trajectories.

Fig. 5.6(e, h) show the estimated wind speed at t=60 min. Comparing these to the ground truth in Fig. 5.6(a), is it clear that both methods estimate the wind field accurately.

In Fig. 5.6(b), we also compare the behavior when using three robots to that of using ten robots. As expected, when there are ten agents the mean clarity is higher (and increases faster) than when there are only three agents.

5.2.6 Conclusions

In conclusion, we have addressed the design of cooperative multi-agent coverage controllers, where the information is shared centrally, but the control decisions are made by each robot independently. We identified a gap between information assimilation algorithms and coverage controllers. Therefore we proposed a method to quantify the value/impact that taking measurements in a domain has on the clarity of our estimate of other parts of the domain. To this end, we utilized Gaussian Processes to model the environment, as well as our earlier work on the clarity dynamics, which in effect quantifies the information gained about the domain due to measurements. We saw that the relative value of measurements is captured by a function S. We used this function to propose two new coverage controllers that, although qualitatively different, still cover the domain and collect information accurately. The

[8] In the interest of space, only the x-component of the speed is shown. Refer to https://github.com/dev10110/multiagent-clarity-based-dynamic-coverage for additional figures.

concepts were demonstrated through a simulation study of collecting information about a wind field.

A key limitation of this work is that we assumed the spatial and temporal hyperparameters of the Gaussian Process were fixed and known a priori. Although a method was described to obtain these hyperparameters from data, in our future work we will aim to develop an online method to estimate the hyperparameters and choose trajectories that improve the quality of the hyperparameters. Finally, it would also be interesting to look into methods to ensure the safety of the robots with a safety constraint that depends on the information collected online. In such a scenario, the objective of collecting information must be weighed against the importance of not violating safety constraints.

Marginal and Conditional Distributions

Consider a random variable $Z \in \mathbb{R}^{n+m}$, given by

$$Z = \begin{bmatrix} X \\ Y \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \mu_x \\ \mu_y \end{bmatrix}, \begin{bmatrix} \Sigma_{xx} & \Sigma_{xy} \\ \Sigma_{yx} & \Sigma_{yy} \end{bmatrix} \right)$$

Then the marginal distributions are given by

$$\begin{aligned} X &\sim \mathcal{N}\left(\mu_x, \Sigma_{xx}\right) \\ Y &\sim \mathcal{N}\left(\mu_y, \Sigma_{yy}\right) \end{aligned}$$

and the conditional distributions are given by

$$\begin{aligned} (X|Y=y) &\sim \mathcal{N}(\mu, \Sigma), \\ \mu &= \mu_x + \Sigma_{xy}\Sigma_{yy}^{-1}\left(y - \mu_y\right) \\ \Sigma &= \Sigma_{xx} - \Sigma_{xy}\Sigma_{yy}^{-1}\Sigma_{yx} \end{aligned}$$

Now consider two random variables X, Y, related by

$$\begin{aligned} X &\sim \mathcal{N}(\mu, P) \\ (Y|X=x) &\sim \mathcal{N}(Cx, R) \end{aligned}$$

where $X \in \mathbb{R}^n$, $Y \in \mathbb{R}^m$, $C \in \mathbb{R}^{m \times n}$, $P \in \mathbb{S}_{++}^n$, $R \in \mathbb{S}_{++}^m$.

What this means is that we have an observation model

$$y = Cx + w, \quad w \sim \mathcal{N}(0, R)$$

Gaussian Processes

The kernel of a GP is defined by the following property

Definition 5.8. *The **kernel function** of a Gaussian Process $Z \sim \mathsf{GP}(m(x), k(x, x'))$ with mean function $m : \mathbb{R}^d \to \mathbb{R}$ and kernel function $k : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ is defined as*

$$k(x, x') = \mathbb{E}\left[(Z(x) - m(x))\,(Z(x') - m(x'))\right].$$

Example 5.5. The ν-th order *Matern kernel* is given by

$$k_\nu(x_1, x_2) = \sigma^2 \frac{2^{1-\nu}}{\Gamma(\nu)} \left(\sqrt{2\nu}\lambda d\right)^\nu K_\nu \left(\sqrt{2\nu}\lambda d\right),$$

where Γ is the gamma function, K_ν is the modified Bessel function of the second kind, $d = \|x_1 - x_2\|$, and $\sigma, \lambda > 0$ are parameters of the kernel. The half-integer Matern kernels are given by

$$\begin{aligned}
k_{1/2}(x_1, x_2) &= \sigma^2 \exp\left(-\lambda d\right) \\
k_{3/2}(x_1, x_2) &= \sigma^2 \left(1 + \sqrt{3}\lambda d\right) \exp\left(-\sqrt{3}\lambda d\right) \\
k_{5/2}(x_1, x_2) &= \sigma^2 \left(1 + \sqrt{5}\lambda d + (5/3)\lambda^2 d^2\right) \exp\left(-\sqrt{5}\lambda d\right)
\end{aligned}$$

where $d = \|x_1 - x_2\|$, and $\sigma, \lambda > 0$ are hyperparameters of the kernel.

Variograms

This section establishes a method to determine the hyperparameters of a Gaussian Process using an Empirical Variogram. This method is significantly more computationally efficient and accurate than standard methods of minimizing the marginal log-likelihood but is only suitable for isotropic kernels.

Consider the data set $D = \{(x_i, y_i)\}_{i=1}^N$, where $x_i \in \mathbb{R}^d, y_i \in \mathbb{R}$. The goal is to determine the parameters of an isotropic kernel $k : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ that best fits the data.

Definition 5.9. *[167, Eq 7.6] The **theoretical variogram** of a stationary random field*

$Z : \mathbb{R}^d \to \mathbb{R}$ with zero mean is

$$\gamma(d) = \frac{1}{2}\mathbb{E}\left[(Z(x') - Z(x))^2\right],$$

where $\|x' - x\| = d$. The expectation is taken over $x, x' \in \mathbb{R}^d$.

The theoretical variogram is related to GP kernels as follows:

Lemma 5.5. *Suppose Z is a zero-mean and isotropic Gaussian Process. Then the kernel $k : \mathbb{R} \to \mathbb{R}$ and the theoretical variogram $\gamma : \mathbb{R} \to \mathbb{R}$ are related by*

$$\gamma(d) = k(0) - k(d)$$

Proof. For brevity, let $f_1 = f(x_1), f_2 = f(x_2)$. By the definition of the kernel, for a zero-mean GP

$$\begin{aligned} k(x_1, x_2) &= E[(f(x_1) - m(x_1))(f(x_2) - m(x_2))] \\ &= E[f_1 f_2] \end{aligned}$$

Similarly, from the definition of the theoretical variogram,

$$\begin{aligned} \gamma(d) &= \frac{1}{2}E[(f(x_1) - f(x_2))^2] \\ &= \frac{1}{2}E[f_1^2] - E[f_1 f_2] + \frac{1}{2}E[f_2^2] \\ &= \frac{1}{2}(E[f_1^2] + E[f_2^2]) - E[f_1 f_2] \\ &= \frac{1}{2}(k(x_1, x_1) + k(x_2, x_2)) - k(x_1, x_2) \\ &= \frac{1}{2}(2k(0)) - k(d) \end{aligned}$$

using $d = x_2 - x_1$. Therefore,

$$\gamma(d) = k(0) - k(d).$$

□

Corollary 5.6. *In the spatiotemporal case, if the kernel is*

$$k(t, x, t', x') = k_t(t, t')k_s(x, x') \tag{5.51}$$

the theoretical variogram is

$$\gamma(d_t, d_s) = k_t(0)k_s(0) - k_t(d_t)k_s(d_s) \tag{5.52}$$

We can use this Lemma to determine the parameters of the kernel. In particular, consider the empirical variogram:

Definition 5.10. *The **empirical semi-variogram** given data D is $\gamma : \mathbb{R} \to \mathbb{R}$,*

$$\gamma(d) = \frac{1}{2|N(d)|} \sum_{N(d)} (y_i - y_j)^2$$

where $N(d) \subset \mathbb{Z} \times \mathbb{Z}$ is the set of pairs (i, j) such that $\|x_i - x_j\| \in (d - \epsilon, d + \epsilon)$ for some $\epsilon > 0$.

Then, given data D, we construct the empirical variogram. For a given kernel k with hyperparameters θ, we can compute the corresponding theoretical variogram, and use least-squares fitting to determine the set of hyperparameters θ that best fit the data D.

5.2.7 Gaussian Processes to Stochastic Differential Equations

This section explains the equivalence between GP and SDE for a class of kernel functions. In this section, we focus on scalar GPs with zero mean,

$$f(t) \sim \mathsf{GP}(0, k(t, t')),$$

where $k : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ is denoted with t to remind the reader that we consider a single (i.e. temporal) dimension.

We use the following convention of a Fourier Transform[9] of a function $g : \mathbb{R} \to \mathbb{R}$:

Definition 5.11. *The Fourier Transform of a function $g : \mathbb{R} \to \mathbb{R}$ is the function $G : \mathbb{R} \to \mathbb{R}$,*

$$G(\omega) = \mathcal{F}[g](\omega) = \int_{-\infty}^{\infty} g(t) e^{-i\omega t} dt$$

*The **Inverse Fourier Transform** is*

$$g(t) = \mathcal{F}^{-1}[G](t) = \frac{1}{2\pi} \int_{-\infty}^{\infty} G(\omega) e^{i\omega t} d\omega$$

[9] In `Mathematica`, one must specify `FourierParameters -> {1, -1}` to yield the correct convention.

This convention has the following properties:

$$\mathcal{F}\left[\frac{d^n g}{dt^n}\right](\omega) = (i\omega)^n \mathcal{F}[g](\omega)$$

The Wiener-Khinchin theorem relates the kernel function to the power spectrum of a stochastic process:

$$S(\omega) = \mathcal{F}[k](\omega)$$

here, we write $k(\tau) = k(t, t')$ for any $|t - t'| = \tau$. When S is a rational function of even order $2n_k$, we can decompose S as

$$S(\omega) = L(\omega)L(-\omega)$$

where

$$L(\omega) = \frac{b_{n_k-1}(i\omega)^{n_k-1} + b_{n_k-2}(i\omega)^{n_k-2} + \cdots + b_0}{(i\omega)^{n_k} + a_{n_k-1}(i\omega)^{n_k-1} + \cdots + a_0}$$

Given this decomposition, we know that the stochastic process f is a realization of a white noise process $W(t)$ that has been colored using the transfer function $L(\omega)$. Therefore, the state-space model of the system is

$$ds = Asdt + BdW$$
$$z = Cs$$

where $s \in \mathbb{R}^{n_k}$ is the state, $W(t)$ is the standard (1D) white noise process. The output $z(t)$ will have the correct kernel function. Here, the constant matrices $A \in \mathbb{R}^{n_k \times n_k}$, $B \in \mathbb{R}^{n_k \times 1}$, $C \in \mathbb{R}^{1 \times n_k}$ are

$$A = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ & & & \ddots & \\ 0 & 0 & 0 & \cdots & 1 \\ -a_0 & -a_1 & -a_2 & \cdots & -a_{n_k-1} \end{bmatrix}, \quad B = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix}$$
$$C = \begin{bmatrix} b_0 & b_1 & b_2 & \cdots & b_{n_k-1} \end{bmatrix}$$

To create a realization of f that has the desired kernel function, simulate the SDE starting

from $s_0 \sim N(0, \Sigma_0)$, where $\Sigma_0 \in \mathbb{R}^{n_k \times n_k}$ is the solution to the Lyapunov equation $AX + XA^T + BB^T = 0$.

Finally, the discrete time version of this, with a sampling period Δt is

$$\begin{aligned} s_{k+1} &= \Phi s_k + w_k, \quad w_k \sim \mathcal{N}(0, W) \\ z_k &= C s_k \end{aligned}$$

where

$$\begin{aligned} \Phi &= e^{A\Delta t} \\ W &= \int_0^{\Delta t} e^{A\tau} B B^T e^{A^T \tau} d\tau \end{aligned}$$

Some analytic expressions are derived below.

Example 5.6 (Matern 1/2). The 1D Matern-1/2 kernel is

$$k_{1/2}(d) = \sigma^2 \exp(-\lambda d)$$

It has a power-spectral density

$$S_{1/2}(\omega) = \frac{2\lambda\sigma^2}{\lambda^2 + \omega^2}$$

and rational decomposition

$$L_{1/2}(\omega) = \frac{\sigma\sqrt{2\lambda}}{(i\omega) + \lambda}$$

Therefore, the state-space representation is

$$\left(\begin{array}{c|c} A & B \\ \hline C & \end{array}\right) = \left(\begin{array}{c|c} -\lambda & 1 \\ \hline \sigma\sqrt{2\lambda} & \end{array}\right)$$

Example 5.7 (Matern 3/2). The 1D Matern-3/2 kernel is

$$k_{3/2}(x_1, x_2) = \sigma^2 \left(1 + \sqrt{3}\lambda d\right) \exp\left(-\sqrt{3}\lambda d\right)$$

It has a power-spectral density

$$S_{3/2}(\omega) = \frac{12\sqrt{3}\lambda^3\sigma^2}{\left(3\lambda^2 + \omega^2\right)^2}$$

and rational decomposition

$$L_{3/2}(\omega) = \frac{\sqrt{12\sqrt{3}}\lambda^{3/2}\sigma}{(i\omega)^2 + 2\sqrt{3}\lambda(i\omega) + 3\lambda^2}$$

Therefore, the state-space representation is

$$\left(\begin{array}{c|c} A & B \\ \hline C & \end{array}\right) = \left(\begin{array}{cc|c} 0 & 1 & 0 \\ -3\lambda^2 & -2\sqrt{3}\lambda & 1 \\ \hline \sqrt{12\sqrt{3}}\lambda^{3/2}\sigma & 0 & \end{array}\right)$$

Example 5.8 (Matern 5/2). The 1D Matern-5/2 kernel is

$$k_{5/2}(x_1, x_2) = \sigma^2 \left(1 + \sqrt{5}\lambda d + (5/3)\lambda^2 d^2\right) \exp\left(-\sqrt{5}\lambda d\right)$$

It has a power-spectral density

$$S_{5/2}(\omega) = \sigma^2 \frac{400\sqrt{5}\lambda^5}{3\left(5\lambda^2 + \omega^2\right)^3}$$

and rational decomposition

$$L_{5/2}(\omega) = \frac{\sqrt{\frac{400\sqrt{5}}{3}}\lambda^{5/2}\sigma}{(i\omega)^3 + 3\sqrt{5}\lambda(i\omega)^2 + 5\sqrt{5}\lambda^3}$$

Therefore, the state-space representation is

$$\left(\begin{array}{c|c} A & B \\ \hline C & \end{array}\right) = \left(\begin{array}{ccc|c} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -5\sqrt{5}\lambda^3 & -15\lambda^2 & -3\sqrt{5}\lambda & 1 \\ \hline \sqrt{\frac{400\sqrt{5}}{3}}\lambda^{5/2}\sigma & 0 & 0 & \end{array}\right)$$

5.2.8 Solutions to the Ricatti Equation

Lemma 5.7. *Consider a differential equation*

$$y'(t) = -\alpha y(t)^2 - \beta y(t) - \gamma \tag{5.53a}$$

$$y(0) = y_0 \tag{5.53b}$$

where $\alpha, \beta, \gamma \in \mathbb{R}$, $\alpha \neq 0$, *and* $\delta^2 = \beta^2 - 4\alpha\gamma > 0$. *Then, the solution is given by*

$$y(t) = \frac{1}{2\alpha}\left(-\beta + \delta + \frac{2\delta\rho_0}{(2\delta + \rho_0)\, e^{\delta t} - \rho_0}\right) \tag{5.54}$$

where $\rho_0 = \beta - \delta + 2\alpha y_0$.

Proof. This is a second-order nonlinear differential equation, also known as the scalar Ricatti equation. As proposed in [82, Ch. 2.15], consider the substitution

$$y(t) = \frac{u'(t)}{\alpha u(t)} \tag{5.55}$$

Then, it is equivalent to the following differential equation

$$u''(t) = -\beta u'(t) - \alpha\gamma u(t) \tag{5.56a}$$

$$y_0 = \frac{u'(0)}{\alpha u(0)} \tag{5.56b}$$

This linear second-order differential equation has a unique solution

$$u(t) = \left(c_2 e^{\delta t} + c_1\right) e^{-\frac{1}{2}t(\beta+\delta)} \tag{5.57}$$

where $\delta = \sqrt{\beta - 4\alpha\gamma}$ and the constants c_1, c_2 depend on the boundary condition. Evaluating the boundary conditions, we have the relationship

$$y_0 = \frac{\frac{1}{2}(c_1 + c_2)(-\beta - \delta) + c_2\delta}{\alpha(c_1 + c_2)} \tag{5.58}$$

Evaluating $y = u'/(\alpha u)$, we have

$$y(t) = \frac{1}{2\alpha}\left(-\beta + \delta - \frac{2c_1\delta}{c_2 e^{\delta t} + c_1}\right) \tag{5.59}$$

Plugging in the boundary condition, we arrive at

$$y(t) = \frac{1}{2\alpha}\left(-\beta + \delta + \frac{2\delta\rho_0}{(2\delta + \rho_0)\, e^{\delta t} - \rho_0}\right) \tag{5.60}$$

where $\rho_0 = \beta - \delta + 2\alpha y_0$, independent of c_1, c_2. □

Corollary 5.8. *The limiting value of* (5.54) *is*

$$y_\infty = \lim_{t\to\infty} y(t) = \frac{\delta - \beta}{2\alpha}. \tag{5.61}$$

Corollary 5.9. *The inverse of* (5.54) *is given by*

$$t = \frac{1}{\delta}\log\left(\frac{\rho_0\,(2\delta + \rho_f)}{\rho_f\,(2\delta + \rho_0)}\right) \tag{5.62}$$

where $\delta^2 = \beta^2 - 4\alpha\gamma$, $\rho_0 = \beta - \delta + 2\alpha y_0$ *and* $\rho_f = \beta - \delta + 2\alpha y_f$.

CHAPTER 6

Conclusions

Contributions

The goal of this thesis was to establish architectures and methodologies for safe autonomous systems. In the introduction, we discussed the top-down vs bottom-up approach to designing safety critical systems, and here we review this idea in the context of the chapters presented above. Recall Figure 1.1, reproduced here:

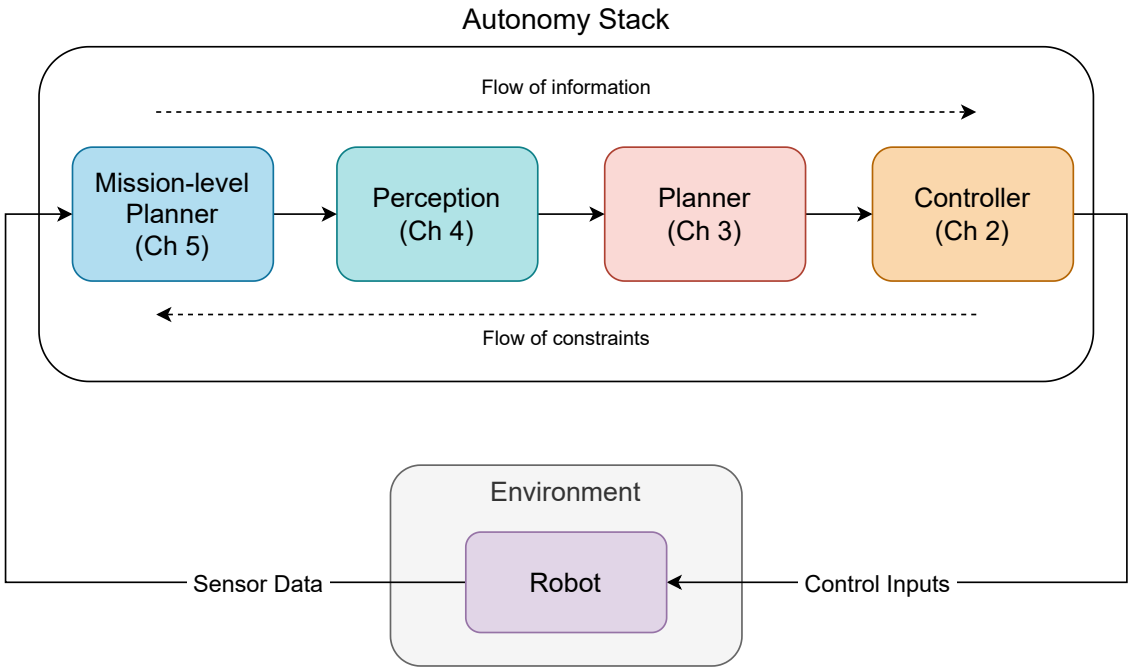

Figure 6.1: A robot's autonomy stack has information flowing from left-to-right, while safety constraints flow from right-to-left. This forwards and backwards flow is a key challenge in designing safety critical autonomous systems.

The dissertation was structured in a bottom-up approach, since this closely resembles the

flow of constraints through an autonomous safety-critical system. The safety guarantees only hold if the controller is able to produce control inputs that lead to safe behavior by the robot. In turn, the controller is only able to produce safe control inputs if it receives a reasonable command from the planner, and if the perception system produces state estimates and obstacle maps that are correct. These requirements of correctness flow from the controller towards the mission level planner and constraints the set of missions that can be executed by the robot.

This also means that for a mission to be successfully executed, the safety constraints need to be flexible enough that the mission can be executed without the safety filters leading the robot astray. This means that although we can and should select the various modules of the autonomy stack to meet mission requirements, we must also design the safety filters such that they do not prevent the mission from occurring.

Of all the results presented in this thesis, we think a few deserve special attention, and especially so since they can be integrated together well:

- In Section 2.3 we proposed a method to synthesize safe controllers in the presence of state estimation uncertainty. This could be used to construct tracking controllers with guarantees that the state remains within a tube of a desired trajectory.
- In Section 3.2 we proposed the `gatekeeper` framework to bridge the planners with controllers in a safety-critical manner. This strategy constructs and updates a committed trajectory based on the information available. If the committed trajectory is tracked by the controller, it guarantees safe operation.
- In Section 4.1 we proposed a method to construct a certifiably correct map of the safe states despite the presence of odometry drift. This produces obstacle maps that can be used by `gatekeeper` to plan safe trajectories.
- In Section 5.1 we proposed and defined the concepts of clarity and perceivability. These allow a system designer to analyze whether the robotic system even has the ability to complete its information gathering task in the first place - this can also be used to modify the actuation or sensing capabilities of the robot to make the robot more efficient at its mission objective.

Limitations and Directions for Future Work

There are a number of limitations of the work presented above, most of which are promising directions for future research. For specific chapters these were discussed earlier in the respective chapters. Here we present focus on the broader impacts that can be developed.

Resilience vs Robustness: Throughout the thesis, the presented controllers and architectures were designed to be robust, not resilient: the proposed methods were robust to disturbances, or bounded errors in various signals, but not resilient, for example to outliers, misinformation, or timing inconsistencies. A system that is resilient is in some sense less fragile in the face of extreme or unlikely events. For a robotic system to operate safely in the real world, we require both robustness and resilience, and new analysis and synthesis tools will need to be developed to address the resilience problem. Naturally, there is a growing literature on this topic, but is beyond the scope of this thesis.

Stochasticity vs Boundedness: In a similar vein, the thesis almost exclusively focused on robustness to bounded disturbances and uncertainties. We made this choice primarily because the mathematical tools to analyze bounded disturbances are far simpler than those for stochastic disturbances and uncertainties. Not only does this introduce conservatism into the control design, the maximum disturbance becomes a crucial tuning parameter that is often not available in practice.

Robustness vs Adaptation: This leads to us to the following question: should the autonomy stack be robust to the worst case disturbance, or adapt to the level of disturbance online? Adapting to the level of disturbance in the system seems like a clear choice, since it should allow system to be aggressive when it is safe to do so, and conservative when there is greater uncertainty. In this thesis however, we focused mainly on the former, again since the analysis is more straightforward - with adaptive methods, one must deeply understand the nature of the incoming data, whether it is descriptive enough to characterize future disturbances, and the closed-loop effect of adapting a controller. These are not trivial questions, and again while a significant amount of literature exists to address this question, more principled analysis will be required before we can deploy general adaptive methods in a safety-critical autonomy stack.

Understanding the Limitations of Sensor Data: As depicted in Figure 6.1, the future trajectory of the robot is determined purely by the control inputs passed into the robot. While this is true, the control inputs are computed from the sensor data produced by the robot in its environment. As control-theorists, we often believe sensor data is perfect, or perhaps corrupted by some noise that is characterized often through Gaussian random variables. However, in modern robotic systems the sensor data is more complicated, including sources like cameras, LIDAR, GPS, and more. This sensor data directly impacts the world model built inside the autonomy stack, and therefore the calculated control inputs. We currently have very limited mathematical tools to analyze the nature of the incoming sensor data, in particular to be able to understand how the quality and uncertainty of the sensor data impacts each of the downstream modules. For example, when using a camera system to

estimate the world, we can use data to identify humans or obstacles in the scene, but we also need methods to understand which parts of the state-space are occluded or have possibly changed dynamically to be able to design control inputs that guarantee safety. While in this example we can use geometric arguments to compute such occlusions, it is not clear how this extends to the case where neural networks predict the obstacles directly from the camera feed: how can one design the neural network to guarantee that it can correctly identify occlusions or dynamic obstacles?[1]

Closing the loop on informative path planning: The connection between the information-driven planning and the rest of the perception, planning, and control modules is still weak. The common paradigm in robotics today is that the perception module can operate independently of the planning module, which can operate independently of the control module. As we have discussed, for safety critical autonomy, this is not the case, and the modules must be co-designed for safety guarantees to hold. However, the connection and dependence on the mission-level planner is still not well understood - can the robotic system be designed such that it determines which information it should collect and how and when it should collect it so that it can perform its mission without deviating from desired plan?

We can construct a simple thought experiment to illustrate this point further. Consider a drone delivery system, tasked with delivering packages in a cluttered urban setting. The wind flow around the buildings it operates near is hard to predict yet crucial for robot to fly safely. In the absence of this information, one can design a safety filter that prevents the robot from getting close to the building, forcing it to fly a longer route. However given the data collected online it is conceivable that the robot can learn the local wind fields and therefore fly closer to the building when it is safe to do so - how and when should the robot deviate from its nominal plan to collect said information? With what confidence does it know this information that it can use this to plan safe trajectories? Is it even valuable to collect said information, and if it is unclear whether this information is useful, how should the robot decide when to make these maneuvers? These are interesting questions not only from the theoretical point of view, but also for practical applications.

A language for architectures:

Finally, I want to return to the idea that the flow-of-information goes from left-to-right, while the flow-of-constraints goes from right-to-left. Although a useful and appealing idea, real robotic systems are rarely this simple and sequential: each module has multiple interdependencies, both at runtime and at design time. Today, each of perception, planning, and control is often handled by separate teams that sometimes have joint meetings (often since

[1]Mark Rober does a great job of illustrating the perils of purely vision-based navigation: https://www.youtube.com/watch?v=IQJL3htsDyQ

they disagree on who is responsible to handle a nasty edge case). I believe this approach is not sustainable into the next generation of more complicated autonomous systems, especially ones that involve large multiagent heterogenous teams of robots that must interact with other autonomous agents. To design such interconnected and interdependent systems, I believe we will need a more abstract and general (mathematical) language to analyze how each module of each agent depends on and influences other modules or other robots. Again, this is an upcoming and promising field of research, but the results there are still in their infancy, since we do not yet know what is the correct abstraction necessary to analyze these autonomy architectures.

BIBLIOGRAPHY

[1] Matthew Abate and Samuel Coogan. Enforcing safety at runtime for systems with disturbances. In *2020 59th IEEE Conference on Decision and Control (CDC)*, pages 2038–2043. IEEE, 2020.

[2] Devansh R Agrawal, Rajiv Govindjee, Taekyung Kim, Trushant Adeshara, Jiangbo Yu, Anurekha Ravikumar, and Dimitra Panagou. Certifiably correct obstacle mapping despite odometry drift (under review). In *2025 Robotic Science and Systems (RSS)*, 2025.

[3] Devansh R Agrawal, Rajiv Govindjee, Jiangbo Yu, Anurekha Ravikumar, and Dimitra Panagou. Online and certifiably correct visual odometry and mapping. *arXiv preprint arXiv:2402.05254*, 2024.

[4] Devansh R Agrawal and Dimitra Panagou. Safe control synthesis via input constrained control barrier functions. In *2021 60th IEEE Conference on Decision and Control (CDC)*, pages 6113–6118. IEEE, 2021.

[5] Devansh R Agrawal and Dimitra Panagou. Safe and robust observer-controller synthesis using control barrier functions. *IEEE Control Systems Letters*, 7:127–132, 2022.

[6] Devansh R Agrawal and Dimitra Panagou. Sensor-based planning and control for robotic systems: Introducing clarity and perceivability. *IEEE Control Systems Letters*, 2023.

[7] Devansh R Agrawal and Dimitra Panagou. Multi-agent clarity-aware dynamic coverage with Gaussian processes. In *IEEE Conference on Decision and Control*, 2024.

[8] Devansh R Agrawal, Hardik Parwana, Ryan K Cosner, Ugo Rosolia, Aaron D Ames, and Dimitra Panagou. A constructive method for designing safe multirate controllers for differentially-flat systems. *IEEE Control Systems Letters*, 6:2138–2143, 2021.

[9] Devansh Ramgopal Agrawal, Ruichang Chen, and Dimitra Panagou. Gatekeeper: Online safety verification and control for nonlinear systems in dynamic environments. In *2023 IEEE IROS*, 2023.

[10] Devansh Ramgopal Agrawal, Ruichang Chen, and Dimitra Panagou. gatekeeper: Online safety verification and control for nonlinear systems in dynamic environments. *IEEE Transactions on Robotics*, 2024.

[11] Mohamadreza Ahmadi, Andrew Singletary, Joel W Burdick, and Aaron D Ames. Safe policy synthesis in multi-agent pomdps via discrete-time barrier functions. In *2019 IEEE 58th Conference on Decision and Control (CDC)*, pages 4797–4803. IEEE, 2019.

[12] Teodoro Alamo, José Manuel Bravo, and Eduardo F Camacho. Guaranteed state estimation by zonotopes. *Automatica*, 41(6):1035–1043, 2005.

[13] Anil Alan, Andrew J Taylor, Chaozhe R He, Gábor Orosz, and Aaron D Ames. Safe controller synthesis with tunable input-to-state safe control barrier functions. *IEEE Control Systems Letters*, 6:908–913, 2021.

[14] A Alessandri. Observer design for nonlinear systems by using input-to-state stability. In *43rd IEEE Conference on Decision and Control (CDC)*, volume 4, pages 3892–3897. IEEE, 2004.

[15] Angelo Alessandri. Lyapunov functions for state observers of dynamic systems using Hamilton–Jacobi inequalities. *Mathematics*, 8(2):202, 2020.

[16] Angelo Alessandri, Patrizia Bagnerini, Mauro Gaggero, and Luca Mantelli. Parameter estimation of fire propagation models using level set methods. *Applied Mathematical Modelling*, 92:731–747, 2021.

[17] F Alizadeh and D Goldfarb. Second-order cone programming. *Mathematical Programming*, 95(1):3–51, 2003.

[18] A. D. Ames, J. W. Grizzle, and P. Tabuada. Control barrier function based quadratic programs with application to adaptive cruise control. In *53rd IEEE Conference on Decision and Control*, pages 6271–6278, 2014.

[19] A. D. Ames, X. Xu, J. W. Grizzle, and P. Tabuada. Control barrier function based quadratic programs for safety critical systems. *IEEE Transactions on Automatic Control*, 62(8):3861–3876, 2017.

[20] Aaron D Ames, Samuel Coogan, Magnus Egerstedt, Gennaro Notomista, Koushil Sreenath, and Paulo Tabuada. Control barrier functions: Theory and applications. In *European Control Conference (ECC)*, pages 3420–3431. IEEE, 2019.

[21] Aaron D Ames, Xiangru Xu, Jessy W Grizzle, and Paulo Tabuada. Control barrier function based quadratic programs for safety critical systems. *IEEE Transactions on Automatic Control*, 62(8):3861–3876, 2017.

[22] Patricia L Andrews. The rothermel surface fire spread model and associated developments: A comprehensive explanation. *Gen. Tech. Rep. RMRS-GTR-371. Fort Collins, CO: US Dept. of Agriculture, Forest Service, Rocky Mountain Research Station. 121 p.*, 371, 2018.

[23] Murat Arcak and Petar Kokotović. Nonlinear observers: a circle criterion design and robustness analysis. *Automatica*, 37(12):1923–1930, 2001.

[24] Andrea Bajcsy, Somil Bansal, Eli Bronstein, Varun Tolani, and Claire J Tomlin. An efficient reachability-based framework for provably safe autonomous navigation in unknown environments. In *2019 IEEE 58th Conference on Decision and Control (CDC)*, pages 1758–1765. IEEE, 2019.

[25] Somil Bansal, Mo Chen, Sylvia Herbert, and Claire J Tomlin. Hamilton-jacobi reachability: A brief overview and recent advances. In *Conference on Decision and Control*, pages 2242–2253. IEEE, 2017.

[26] Timothy D Barfoot. *State estimation for robotics*. Cambridge University Press, 2024.

[27] William Bentz. *Dynamic Coverage Control and Estimation in Collaborative Networks of Human-Aerial/Space Co-Robots*. PhD thesis, University of Michigan, Ann Arbor, 2020.

[28] William Bentz and Dimitra Panagou. A hybrid approach to persistent coverage in stochastic environments. *Automatica*, 109:108554, 2019.

[29] Andreas Bernard. *Lifted: A cultural history of the elevator*. New York University Press, 2014.

[30] Pauline Bernard, Vincent Andrieu, and Daniele Astolfi. Observer design for continuous-time dynamical systems. *Annual Reviews in Control*, 2022.

[31] Franco Blanchini. Set invariance in control. *Automatica*, 35(11):1747–1767, 1999.

[32] Franco Blanchini, Stefano Miani, et al. *Set-theoretic methods in control*. Springer, Birkhäuser Cham, 2015.

[33] Grigoriy Blekherman, Pablo A Parrilo, and Rekha R Thomas. *Semidefinite optimization and convex algebraic geometry*. SIAM, 2012.

[34] Francesco Borrelli, Alberto Bemporad, and Manfred Morari. *Predictive control for linear and hybrid systems*. Cambridge University Press, 2017.

[35] Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[36] Joseph Breeden, Kunal Garg, and Dimitra Panagou. Control barrier functions in sampled-data systems. *IEEE Control Systems Letters*, 6:367–372, 2021.

[37] Joseph Breeden and Dimitra Panagou. Guaranteed safe spacecraft docking with control barrier functions. *IEEE Control Systems Letters*, 6:2000–2005, 2021.

[38] Lukas Brunke, Melissa Greeff, Adam W Hall, Zhaocong Yuan, Siqi Zhou, Jacopo Panerati, and Angela P Schoellig. Safe learning in robotics: From learning-based control to safe reinforcement learning. *Annual Review of Control, Robotics, and Autonomous Systems*, 5(1):411–444, 2022.

[39] Adam Bry and Nicholas Roy. Rapidly-exploring random belief trees for motion planning under uncertainty. In *IEEE ICRA*, pages 723–730, 2011.

[40] Arthur Earl Bryson. *Applied optimal control: optimization, estimation and control*. CRC Press, 1975.

[41] Jonas Buchli, Mrinal Kalakrishnan, Michael Mistry, Peter Pastor, and Stefan Schaal. Compliant quadruped locomotion over rough terrain. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 814–820, 2009.

[42] Chenghui Cai and Silvia Ferrari. Information-driven sensor path planning by approximate cell decomposition. *IEEE TSMC*, 39(3):672–689, 2009.

[43] Nannan Cao, Kian Hsiang Low, and John M Dolan. Multi-robot informative path planning for active sensing of environmental phenomena: A tale of two algorithms. *arXiv preprint arXiv:1302.0723*, 2013.

[44] Andrea Carron, Marco Todescato, Ruggero Carli, Luca Schenato, and Gianluigi Pillonetto. Machine learning meets Kalman filtering. In *55th IEEE Conference on Decision and Control*, pages 4594–4599, 2016.

[45] Kenny Chen, Ryan Nemiroff, and Brett T Lopez. Direct lidar-inertial odometry: Lightweight lio with continuous-time motion correction. In *IEEE ICRA*, pages 3983–3989. IEEE, 2023.

[46] Mo Chen, Sylvia L Herbert, Haimin Hu, Ye Pu, Jaime Fernandez Fisac, Somil Bansal, SooJean Han, and Claire J Tomlin. Fastrack: a modular framework for real-time motion planning and guaranteed safe tracking. *IEEE Transactions on Automatic Control*, 66(12):5861–5876, 2021.

[47] Yuxiao Chen, Mrdjan Jankovic, Mario Santillo, and Aaron D Ames. Backup control barrier functions: Formulation and comparative study. In *2021 60th IEEE Conference on Decision and Control*, pages 6835–6841, 2021.

[48] Jason J Choi, Donggun Lee, Koushil Sreenath, Claire J Tomlin, and Sylvia L Herbert. Robust control barrier–value functions for safety-critical control. In *2021 60th IEEE Conference on Decision and Control (CDC)*, pages 6814–6821. IEEE, 2021.

[49] Glen Chou. *Safe End-to-end Learning-based Robot Autonomy via Integrated Perception, Planning, and Control*. PhD thesis, University of Michigan, Ann Arbor, 2022.

[50] Ryan B Christianson, Ryan M Pollyea, and Robert B Gramacy. Traditional kriging versus modern Gaussian processes for large-scale mining data. *Statistical Analysis and Data Mining: The ASA Data Science Journal*, 16(5):488–506, 2023.

[51] Andrew Clark. Control barrier functions for complete and incomplete information stochastic systems. In *American Control Conference (ACC)*, pages 2928–2935. IEEE, 2019.

[52] Jorge Cortes, Sonia Martinez, Timur Karatas, and Francesco Bullo. Coverage control for mobile sensing networks. *IEEE T-RO*, 20(2):243–255, 2004.

[53] Wenceslao Shaw Cortez and Dimos V Dimarogonas. Correct-by-design control barrier functions for euler-lagrange systems with input constraints. In *American Control Conference (ACC)*, pages 950–955. IEEE, 2020.

[54] Miguel G Cruz and Martin E Alexander. The 10% wind speed rule of thumb for estimating a wildfire's forward rate of spread in forests and shrublands. *Annals of Forest Science*, 76(2):1–11, 2019.

[55] Charles Dawson, Bethany Lowenkamp, Dylan Goff, and Chuchu Fan. Learning safe, generalizable perception-based hybrid control with certificates. *IEEE Robotics and Automation Letters*, 7(2):1904–1911, 2022.

[56] Charles Dawson, Zengyi Qin, Sicun Gao, and Chuchu Fan. Safe nonlinear control using robust neural lyapunov-barrier functions. In *Conference on Robot Learning*, pages 1724–1735. PMLR, 2022.

[57] Sarah Dean, Andrew Taylor, Ryan Cosner, Benjamin Recht, and Aaron Ames. Guaranteeing safety of learned perception modules via measurement-robust control barrier functions. In *Conference on Robot Learning*, pages 654–670. PMLR, 2021.

[58] Jean-Charles Delvenne and Henrik Sandberg. Towards a thermodynamics of control: Entropy, Energy and Kalman filtering. In *IEEE CDC*, pages 3109–3114, 2013.

[59] Dayi Dong, Henry Berger, and Ian Abraham. Time optimal ergodic search. *arXiv preprint arXiv:2305.11643*, 2023.

[60] Louis Dressel and Mykel J Kochenderfer. Tutorial on the generation of ergodic trajectories with projection-based gradient descent. *IET Cyber-Physical Systems: Theory & Applications*, 4(2):89–100, 2019.

[61] Chuchu Fan, Kristina Miller, and Sayan Mitra. Fast and guaranteed safe controller synthesis for nonlinear vehicle models. In *International Conference on Computer Aided Verification*, pages 629–652. Springer, 2020.

[62] Michel Fliess, Jean Lévine, Philippe Martin, and Pierre Rouchon. Flatness and defect of non-linear systems: introductory theory and examples. *International journal of control*, 61(6):1327–1361, 1995.

[63] Michel Fliess, Jean Lévine, Philippe Martin, and Pierre Rouchon. A lie-backlund approach to equivalence and flatness of nonlinear systems. *IEEE Transactions on automatic control*, 44(5):922–937, 1999.

[64] Komei Fukuda and Alain Prodon. Double description method revisited. In *Franco-Japanese and Franco-Chinese conference on combinatorics and computer science*, pages 91–111. Springer, 1995.

[65] Fadri Furrer, Michael Burri, Markus Achtelik, and Roland Siegwart. *Robot Operating System (ROS): The Complete Reference (Volume 1)*, chapter RotorS—A Modular Gazebo MAV Simulator Framework, pages 595–625. Springer International Publishing, Cham, 2016.

[66] Yiqi Gao, Andrew Gray, H Eric Tseng, and Francesco Borrelli. A tube-based robust nonlinear predictive control approach to semiautonomous ground vehicles. *Vehicle System Dynamics*, 52(6):802–823, 2014.

[67] K. Garg and D. Panagou. Control-lyapunov and control-barrier functions based quadratic program for spatio-temporal specifications. In *2019 IEEE 58th Conference on Decision and Control (CDC)*, pages 1422–1429, 2019.

[68] Kunal Garg, Ryan K Cosner, Ugo Rosolia, Aaron D Ames, and Dimitra Panagou. Multi-rate control design under input constraints via fixed-time barrier functions. *Control Systems Letters*, 2021.

[69] Kunal Garg, James Usevitch, Joseph Breeden, Mitchell Black, Devansh Agrawal, Hardik Parwana, and Dimitra Panagou. Advances in the theory of control barrier functions: Addressing practical challenges in safe control synthesis for autonomous and robotic systems. *Annual Reviews in Control*, 57:100945, 2024.

[70] Arthur Gelb et al. *Applied optimal estimation*. MIT press, 1974.

[71] T. Gurriet, A. Singletary, J. Reher, L. Ciarletta, E. Feron, and A. Ames. Towards a framework for realizable safety critical control through active set invariance. In *2018 ACM/IEEE 9th International Conference on Cyber-Physical Systems (ICCPS)*, pages 98–106, 2018.

[72] Thomas Gurriet, Mark Mote, Andrew Singletary, Eric Feron, and Aaron D Ames. A scalable controlled set invariance framework with practical safety guarantees. In *2019 IEEE CDC*, pages 2046–2053. IEEE, 2019.

[73] Thomas Gurriet, Petter Nilsson, Andrew Singletary, and Aaron D Ames. Realizable set invariance conditions for cyber-physical systems. In *2019 IEEE ACC*, pages 3642–3649. IEEE, 2019.

[74] Thomas Gurriet, Andrew Singletary, Jacob Reher, Laurent Ciarletta, Eric Feron, and Aaron Ames. Towards a framework for realizable safety critical control through active set invariance. In *International Conference on Cyber-Physical Systems*, pages 98–106. IEEE, 2018.

[75] William W Hager. Lipschitz continuity for constrained processes. *SIAM Journal on Control and Optimization*, 17(3):321–338, 1979.

[76] Daniel Harabor and Alban Grastien. Online graph pruning for pathfinding on grid maps. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 25, pages 1114–1119, 2011.

[77] Ben Haydon, Kirti D Mishra, Patrick Keyantuo, Dimitra Panagou, Fotini Chow, Scott Moura, and Chris Vermillion. Dynamic coverage meets regret: Unifying two control performance measures for mobile agents in spatiotemporally varying environments. In *IEEE CDC*, pages 521–526, 2021.

[78] Sylvia L Herbert, Mo Chen, SooJean Han, Somil Bansal, Jaime F Fisac, and Claire J Tomlin. Fastrack: A modular framework for fast and guaranteed safe motion planning. In *Conference on Decision and Control*, pages 1517–1522. IEEE, 2017.

[79] Geoffrey A Hollinger and Gaurav S Sukhatme. Sampling-based motion planning for robotic information gathering. In *Robotics: Science and Systems*, volume 3, pages 1–8, 2013.

[80] Armin Hornung, Kai M Wurm, Maren Bennewitz, Cyrill Stachniss, and Wolfram Burgard. Octomap: An efficient probabilistic 3d mapping framework based on octrees. *Autonomous robots*, 34:189–206, 2013.

[81] Adam Howell and J Karl Hedrick. Nonlinear observer design via convex optimization. In *2002 American Control Conference (ACC)*, volume 3, pages 2088–2093. IEEE, 2002.

[82] Edward L Ince. *Ordinary differential equations*. Longmans, GReen and Company Limited, 1927.

[83] Niloofar Jahanshahi, Pushpak Jagtap, and Majid Zamani. Synthesis of stochastic systems with partial information via control barrier functions. *IFAC-PapersOnLine*, 53(2):2441–2446, 2020.

[84] Mrdjan Jankovic. Robust control barrier functions for constrained stabilization of nonlinear systems. *Automatica*, 96:359–367, 2018.

[85] Luc Jaulin. Nonlinear bounded-error state estimation of continuous-time systems. *Automatica*, 38(6):1079–1082, 2002.

[86] Bomin Jiang, Zhiyong Sun, and Brian DO Anderson. Higher order Voronoi based mobile coverage control. In *IEEE ACC*, pages 1457–1462, 2015.

[87] Yutaka Kanayama, Yoshihiko Kimura, Fumio Miyazaki, and Tetsuo Noguchi. A stable tracking control method for an autonomous mobile robot. In *IEEE ICRA*, pages 384–389. IEEE, 1990.

[88] Sertac Karaman, Matthew R Walter, Alejandro Perez, Emilio Frazzoli, and Seth Teller. Anytime motion planning using the rrt. In *2011 IEEE ICRA*, pages 1478–1483. IEEE, 2011.

[89] Peter Karkus, Xiao Ma, David Hsu, Leslie Pack Kaelbling, Wee Sun Lee, and Tomás Lozano-Pérez. Differentiable algorithm networks for composable robot learning. *arXiv preprint arXiv:1905.11602*, 2019.

[90] Christopher M Kellett. A compendium of comparison function results. *Mathematics of Control, Signals, and Systems*, 26(3):339–374, 2014.

[91] Hassan Khalil. *Nonlinear control*, volume 406. Pearson, Boston, 2015.

[92] Hassan K Khalil. Nonlinear systems, 3rd edition. *Prentice Hall*, 2002.

[93] Markus Kögel and Rolf Findeisen. Discrete-time robust model predictive control for continuous-time nonlinear systems. In *American Control Conference*, pages 924–930. IEEE, 2015.

[94] Johannes Köhler, Raffaele Soloperto, Matthias A Müller, and Frank Allgöwer. A computationally efficient robust model predictive control framework for uncertain nonlinear systems. *IEEE Transactions on Automatic Control*, 66(2):794–801, 2020.

[95] Shreyas Kousik, Sean Vaskov, Fan Bu, Matthew Johnson-Roberson, and Ram Vasudevan. Bridging the gap between safety and real-time performance in receding-horizon trajectory design for mobile robots. *The International Journal of Robotics Research*, 39(12):1419–1469, 2020.

[96] Armin Küper and Steffen Waldherr. Numerical Gaussian process Kalman filtering for spatiotemporal systems. *IEEE Transactions on Automatic Control*, 68(5):3131–3138, 2022.

[97] Alan Laub. A schur method for solving algebraic riccati equations. *IEEE Transactions on automatic control*, 24(6):913–921, 1979.

[98] Steven M LaValle. *Planning algorithms*. Cambridge university press, 2006.

[99] Will Lavanakul, Jason J Choi, Koushil Sreenath, and Claire J Tomlin. Safety filters for black-box dynamical systems by learning discriminating hyperplanes. *arXiv preprint arXiv:2402.05279*, 2024.

[100] Taeyoung Lee, Melvin Leok, and N Harris McClamroch. Geometric tracking control of a quadrotor uav on se (3). In *49th IEEE CDC*, pages 5420–5425. IEEE, 2010.

[101] Benoît Legat. Polyhedral computation. In *JuliaCon*, July 2023.

[102] Jean Lévine. On the equivalence between differential flatness and dynamic feedback linearizability. *IFAC Proceedings Volumes*, 40(20):338–343, 2007.

[103] Jean Levine. *Analysis and control of nonlinear systems: A flatness-based approach*. Springer Science & Business Media, 2009.

[104] Frank L Lewis. *Applied optimal control and estimation*. Prentice Hall PTR, 1992.

[105] Frank L Lewis, Lihua Xie, and Dan Popa. *Optimal and robust estimation: with an introduction to stochastic control theory*. CRC press, 2017.

[106] Junbin Liang, Ming Liu, and Xiaoyan Kui. A survey of coverage problems in wireless sensor networks. *Sensors & Transducers*, 163(1):240, 2014.

[107] Sikang Liu, Michael Watterson, Kartik Mohta, Ke Sun, Subhrajit Bhattacharya, Camillo J Taylor, and Vijay Kumar. Planning dynamically feasible trajectories for quadrotors using safe flight corridors in 3-d complex environments. *IEEE RAL*, 2(3):1688–1695, 2017.

[108] Simin Liu, Changliu Liu, and John Dolan. Safe control under input limits with neural control barrier functions. In *Conference on Robot Learning*, pages 1970–1980. PMLR, 2023.

[109] Christian Llanes, Matthew Abate, and Samuel Coogan. Safety from in-the-loop reachability for cyber-physical systems. In *Proceedings of the Workshop on Computation-Aware Algorithmic Design for Cyber-Physical Systems*, pages 9–10, 2021.

[110] Brett T Lopez and Jonathan P How. Aggressive collision avoidance with limited field-of-view sensing. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1358–1365, 2017.

[111] Joshua G Mangelson, Maani Ghaffari, Ram Vasudevan, and Ryan M Eustice. Characterizing the uncertainty of jointly distributed poses in the lie algebra. *IEEE Transactions on Robotics*, 36(5):1371–1388, 2020.

[112] Roman Marchant and Fabio Ramos. Bayesian optimisation for informative continuous path planning. In *IEEE ICRA*, pages 6136–6143, 2014.

[113] Matteo Marchi, Jonathan Bunton, Bahman Gharesifard, and Paulo Tabuada. LiDAR point cloud registration with formal guarantees. In *IEEE CDC*, pages 3462–3467, 2022.

[114] Phillipe Martin, Richard M Murray, and Pierre Rouchon. Flat systems, equivalence and trajectory generation. *CDS Technical Report*, 2003.

[115] George Mathew and Igor Mezić. Metrics for ergodicity and design of ergodic dynamics for multi-agent systems. *Physica D: Nonlinear Phenomena*, 240(4-5):432–442, 2011.

[116] Daniel Mellinger and Vijay Kumar. Minimum snap trajectory generation and control for quadrotors. In *IEEE International Conference on Robotics and Automation*, pages 2520–2525. IEEE, 2011.

[117] Roxane Merat, Giovanni Cioffi, Leonard Bauersfeld, and Davide Scaramuzza. Drift-free visual slam using digital twins. *IEEE Robotics and Automation Letters*, 10(2):1633–1640, 2025.

[118] Alexander Millane, Helen Oleynikova, Emilie Wirbel, Remo Steiner, Vikram Ramasamy, David Tingdahl, and Roland Siegwart. nvblox: Gpu-accelerated incremental signed distance field mapping, 2024.

[119] I. M. Mitchell, A. M. Bayen, and C. J. Tomlin. A time-dependent Hamilton-Jacobi formulation of reachable sets for continuous dynamic games. *IEEE Transactions on Automatic Control*, 50(7):947–957, 2005.

[120] Ian M Mitchell and Jeremy A Templeton. A toolbox of Hamilton-Jacobi solvers for analysis of nondeterministic continuous and hybrid systems. In *Hybrid Systems: Computation and Control.*, pages 480–494. Springer, 2005.

[121] Brady Moon, Satrajit Chatterjee, and Sebastian Scherer. Tigris: An informed sampling-based algorithm for informative path planning. In *IEEE IROS*, pages 5760–5766, 2022.

[122] Benjamin Morris, Matthew J Powell, and Aaron D Ames. Sufficient conditions for the lipschitz continuity of qp-based multi-objective control of humanoid robots. In *52nd IEEE Conference on Decision and Control*, pages 2920–2926. IEEE, 2013.

[123] Mark W Mueller, Michael Hamer, and Raffaello D'Andrea. Fusing ultra-wideband range measurements with accelerometers and rate gyroscopes for quadrocopter state estimation. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1730–1736, May 2015.

[124] Richard M Murray, Muruhan Rathinam, and Willem Sluis. Differential flatness of mechanical control systems: A catalog of prototype systems. In *ASME international mechanical engineering congress and exposition*. Citeseer, 1995.

[125] Mitio Nagumo. Über die lage der integralkurven gewöhnlicher differentialgleichungen. *Proceedings of the physico-mathematical society of Japan. 3rd Series*, 24:551–559, 1942.

[126] Kaleb Ben Naveed, Devansh Agrawal, Christopher Vermillion, and Dimitra Panagou. Eclares: Energy-aware clarity-driven ergodic search. In *IEEE International Conference on Robotics and Automation*, pages 14326–14332, 2024.

[127] Sema Oktug, Anar Khalilov, and Hakan Tezcan. 3d coverage analysis under heterogeneous deployment strategies in wireless sensor networks. In *2008 Fourth Advanced International Conference on Telecommunications*, pages 199–204. IEEE, 2008.

[128] Helen Oleynikova, Zachary Taylor, Marius Fehr, Roland Siegwart, and Juan Nieto. Voxblox: Incremental 3d euclidean signed distance fields for on-board mav planning. In *2017 IEEE/RSJ IROS*, pages 1366–1373. IEEE, 2017.

[129] Dimitra Panagou, Dušan M Stipanović, and Petros G Voulgaris. Distributed dynamic coverage and avoidance control under anisotropic sensing. *IEEE TCNS*, 4(4):850–862, 2016.

[130] H. Park, S. Di Cairano, and I. Kolmanovsky. Model predictive control for spacecraft rendezvous and docking with a rotating/tumbling platform and for debris avoidance. In *Proceedings of the 2011 American Control Conference*, pages 1922–1927, 2011.

[131] Marija Popović, Teresa Vidal-Calleja, Jen Jen Chung, Juan Nieto, and Roland Siegwart. Informative path planning for active field mapping under localization uncertainty. In *IEEE ICRA*, pages 10751–10757, 2020.

[132] Stephen Prajna. Barrier certificates for nonlinear model validation. *Automatica*, 42(1):117–126, 2006.

[133] Dimitrios Pylorof, Efstathios Bakolas, and Kevin S. Chan. Design of robust Lyapunov-based observers for nonlinear systems with sum-of-squares programming. *IEEE Control Systems Letters*, 4(2):283–288, 2020.

[134] Christopher Rackauckas and Qing Nie. `Differentialequations.jl`–a performant and feature-rich ecosystem for solving differential equations in `julia`. *J. Open Research Software*, 5(1), 2017.

[135] Suresh Ramasamy, Guofan Wu, and Koushil Sreenath. Dynamically feasible motion planning through partial differential flatness. In *Robotics: Science and Systems*. Citeseer, 2014.

[136] James Blake Rawlings, David Q Mayne, and Moritz Diehl. *Model predictive control: theory, computation, and design*, volume 2. Nob Hill Publishing Madison, WI, 2017.

[137] Konrad Reif, Frank Sonnemann, and Rolf Unbehauen. An EKF-Based Nonlinear Observer with a Prescribed Degree of Stability. *Automatica*, 34(9):1119–1123, 1998.

[138] Charles Richter, Adam Bry, and Nicholas Roy. Polynomial trajectory planning for aggressive quadrotor flight in dense indoor environments. In *16th Int. Symposium on Robotics Research*, pages 649–666. Springer, 2016.

[139] David M Rosen, Luca Carlone, Afonso S Bandeira, and John J Leonard. SE-Sync: a certifiably correct algorithm for synchronization over the special euclidean group. *IEEE IJRR*, 38(2-3):95–125, 2019.

[140] Antoni Rosinol, Marcus Abate, Yun Chang, and Luca Carlone. Kimera: an open-source library for real-time metric-semantic localization and mapping. In *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2020.

[141] Antoni Rosinol, John J Leonard, and Luca Carlone. Nerf-slam: Real-time dense monocular slam with neural radiance fields. In *IEEE ICRA*, pages 3437–3444. IEEE, 2023.

[142] Ugo Rosolia and Aaron D Ames. Multi-rate control design leveraging control barrier functions and model predictive control policies. *IEEE Control Systems Letters*, 5(3):1007–1012, 2020.

[143] Ugo Rosolia, Ashwin Carvalho, and Francesco Borrelli. Autonomous racing using learning model predictive control. In *2017 American Control Conference (ACC)*, pages 5115–5120. IEEE, 2017.

[144] Richard C Rothermel. *A mathematical model for predicting fire spread in wildland fuels*, volume 115. Intermountain Forest & Range Experiment Station, Forest Service, US . . . , 1972.

[145] Simo Sarkka and Jouni Hartikainen. Infinite-dimensional Kalman filtering approach to spatio-temporal Gaussian process regression. In *Artificial intelligence and statistics*, pages 993–1001. PMLR, 2012.

[146] Simo Särkkä and Arno Solin. *Applied stochastic differential equations*, volume 10. Cambridge University Press, 2019.

[147] Davide Scaramuzza and Friedrich Fraundorfer. Visual odometry [tutorial]. *IEEE Robot. & Automat. Mag.*, 18(4):80–92, 2011.

[148] Christoph Schlager, Gottfried Kirchengast, and Jürgen Fuchsberger. Generation of high-resolution wind fields from the wegenernet dense meteorological station network in southeastern austria. *Weather and Forecasting*, 32(4):1301–1319, 2017.

[149] Claude Elwood Shannon. A mathematical theory of communication. *The Bell system technical journal*, 27(3):379–423, 1948.

[150] Hyungbo Shim and Daniel Liberzon. Nonlinear observers robust to measurement disturbances in an iss sense. *IEEE Trans. on Automatic Control*, 61(1):48–61, 2015.

[151] Sumeet Singh, Mo Chen, Sylvia L Herbert, Claire J Tomlin, and Marco Pavone. Robust tracking with model mismatch for fast and safe planning: an sos optimization approach. In *Algorithmic Foundations of Robotics XIII: Proceedings of the 13th Workshop on the Algorithmic Foundations of Robotics 13*, pages 545–564. Springer, 2020.

[152] Andrew Singletary, Yuxiao Chen, and Aaron D Ames. Control barrier functions for sampled-data systems with input delays. In *Conference on Decision and Control*, pages 804–809. IEEE, 2020.

[153] Andrew Singletary, Aiden Swann, Yuxiao Chen, and Aaron D Ames. Onboard safety guarantees for racing drones: High-speed geofencing with control barrier functions. *IEEE RAL*, 7(2):2897–2904, 2022.

[154] Oswin So, Zachary Serlin, Makai Mann, Jake Gonzales, Kwesi Rutledge, Nicholas Roy, and Chuchu Fan. How to train your neural control barrier function: Learning safety filters for complex input-constrained systems. *arXiv preprint arXiv:2310.15478*, 2023.

[155] Joan Sola, Jeremie Deray, and Dinesh Atchuthan. A micro lie theory for state estimation in robotics. *arXiv preprint arXiv:1812.01537*, 2018.

[156] Eduardo D Sontag. Input to state stability: Basic concepts and results. In *Nonlinear and optimal control theory*, pages 163–220. Springer, 2008.

[157] B. Stellato, G. Banjac, P. Goulart, A. Bemporad, and S. Boyd. OSQP: an operator splitting solver for quadratic programs. *Mathematical Programming Computation*, 12(4):637–672, 2020.

[158] Julian Straub, Thomas Whelan, Lingni Ma, Yufan Chen, Erik Wijmans, Simon Green, Jakob J. Engel, Raul Mur-Artal, Carl Ren, Shobhit Verma, Anton Clarkson, Mingfei Yan, Brian Budge, Yajie Yan, Xiaqing Pan, June Yon, Yuyang Zou, Kimberly Leon, Nigel Carter, Jesus Briales, Tyler Gillingham, Elias Mueggler, Luis Pesqueira, Manolis Savva, Dhruv Batra, Hauke M. Strasdat, Renzo De Nardi, Michael Goesele, Steven Lovegrove, and Richard Newcombe. The Replica dataset: A digital replica of indoor spaces. *arXiv preprint arXiv:1906.05797*, 2019.

[159] Ezra Tal and Sertac Karaman. Accurate tracking of aggressive quadrotor trajectories using incremental nonlinear dynamic inversion and differential flatness. *IEEE Transactions on Control Systems Technology*, 29(3):1203–1218, 2020.

[160] MTCAJ Thomas and A Thomas Joy. *Elements of information theory*. Wiley-Interscience, 2006.

[161] Yulun Tian, Yun Chang, Fernando Herrera Arias, Carlos Nieto-Granda, Jonathan P How, and Luca Carlone. Kimera-multi: Robust, distributed, dense metric-semantic slam for multi-robot systems. *IEEE TRO*, 38(4), 2022.

[162] Sander Tonkens and Sylvia Herbert. Refining control barrier functions through Hamilton-Jacobi reachability. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 13355–13362, 2022.

[163] Jesus Tordesillas and Jonathan P How. Mader: Trajectory planner in multiagent and dynamic environments. *IEEE TRO*, 38(1):463–476, 2021.

[164] Jesus Tordesillas, Brett T Lopez, Michael Everett, and Jonathan P How. Faster: Fast and safe trajectory planner for navigation in unknown environments. *IEEE TRO*, 38(2):922–938, 2021.

[165] Hugo Touchette and Seth Lloyd. Information-theoretic approach to the study of control systems. *Physica A: Statistical Mechanics and its Applications*, 331(1-2):140–172, 2004.

[166] Henk J Van Waarde, Jaap Eising, Harry L Trentelman, and M Kanat Camlibel. Data informativity: a new perspective on data-driven analysis and control. *IEEE TAC*, 65(11):4753–4768, 2020.

[167] Hans Wackernagel. *Multivariate geostatistics: an introduction with applications*. Springer Science & Business Media, 2003.

[168] Li Wang, Aaron D Ames, and Magnus Egerstedt. Safe certificate-based maneuvers for teams of quadrotors using differential flatness. In *International Conference on Robotics and Automation (ICRA)*, pages 3293–3298. IEEE, 2017.

[169] Dustin J Webb and Jur Van Den Berg. Kinodynamic RRT*: Asymptotically optimal motion planning for robots with linear dynamics. In *2013 IEEE ICRA*, pages 5054–5061, 2013.

[170] Peter Wieland and Frank Allgöwer. Constructive safety using control barrier functions. *IFAC Proceedings Volumes*, 40(12):462–467, 2007.

[171] Christopher KI Williams and Carl Edward Rasmussen. *Gaussian processes for machine learning*, volume 2. MIT press Cambridge, MA, 2006.

[172] Albert Wu, Sadra Sadraddini, and Russ Tedrake. R3t: Rapidly-exploring random reachable set tree for optimal kinodynamic planning of nonlinear hybrid systems. In *International Conference on Robotics and Automation*, pages 4245–4251. IEEE, 2020.

[173] Guofan Wu and Koushil Sreenath. Safety-critical and constrained geometric control synthesis using control lyapunov and control barrier functions for systems evolving on manifolds. In *2015 American Control Conference (ACC)*, pages 2038–2044. IEEE, 2015.

[174] Guofan Wu and Koushil Sreenath. Safety-critical control of a planar quadrotor. In *American control conference (ACC)*, pages 2252–2258. IEEE, 2016.

[175] Chenxi Xiao and Juan Wachs. Nonmyopic informative path planning based on global kriging variance minimization. *IEEE RAL*, 7(2):1768–1775, 2022.

[176] Wei Xiao and Calin Belta. Control barrier functions for systems with high relative degree. In *IEEE 58th Conference on Decision and Control (CDC)*, pages 474–479. IEEE, 2019.

[177] X. Xu, J. W. Grizzle, P. Tabuada, and A. D. Ames. Correctness guarantees for the composition of lane keeping and adaptive cruise control. *IEEE Transactions on Automation Science and Engineering*, 15(3):1216–1229, 2018.

[178] Xiangru Xu, Paulo Tabuada, Jessy W. Grizzle, and Aaron D. Ames. Robustness of control barrier functions for safety critical control. *IFAC-PapersOnLine*, 48(27):54–61, 2015. Analysis and Design of Hybrid Systems ADHS.

[179] Heng Yang, Jingnan Shi, and Luca Carlone. TEASER: Fast and certifiable point cloud registration. *IEEE TRO.*, 37(2):314–333, 2020.

[180] He Yin, Monimoy Bujarbaruah, Murat Arcak, and Andrew Packard. Optimization based planner–tracker design for safety guarantees. In *American Control Conference*, pages 5194–5200. IEEE, 2020.

[181] Shuyou Yu, Christoph Maier, Hong Chen, and Frank Allgöwer. Tube mpc scheme based on robust control invariant set with application to lipschitz nonlinear systems. *Systems & Control Letters*, 62(2):194–200, 2013.

[182] Zhelin Yu, Lidong Zhu, and Guoyu Lu. Vins-motion: tightly-coupled fusion of vins and motion constraint. In *IEEE ICRA*, pages 7672–7678. IEEE, 2021.

[183] Ji Zhang and Sanjiv Singh. Ins assisted monocular visual odometry for aerial vehicles. In *Field and Service Robotics: Results of the 9th International Conference*, pages 183–197. Springer, 2015.

[184] Zichao Zhang and Davide Scaramuzza. A tutorial on quantitative trajectory evaluation for visual (-inertial) odometry. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 7244–7251. IEEE, 2018.

[185] Zichao Zhang and Davide Scaramuzza. Beyond point clouds: Fisher information field for active visual localization. In *IEEE ICRA*, pages 5986–5992, 2019.

[186] Boyu Zhou, Jie Pan, Fei Gao, and Shaojie Shen. Raptor: Robust and perception-aware trajectory replanning for quadrotor fast flight. *IEEE TRO*, 37(6):1992–2009, 2021.

[187] Boyu Zhou, Yichen Zhang, Xinyi Chen, and Shaojie Shen. FUEL: Fast UAV exploration using incremental frontier structure and hierarchical planning. *IEEE RAL*, 6(2):779–786, 2021.

[188] Zihan Zhu, Songyou Peng, Viktor Larsson, Weiwei Xu, Hujun Bao, Zhaopeng Cui, Martin R. Oswald, and Marc Pollefeys. Nice-slam: Neural implicit scalable encoding for slam. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.