UC Berkeley
UC Berkeley Electronic Theses and Dissertations

Title
Game-Theoretic Safety Assurance for Human-Centered Robotic Systems

Permalink
https://escholarship.org/uc/item/1xh7s61r

Author
Fernandez Fisac, Jaime

Publication Date
2019

Peer reviewed|Thesis/dissertation

eScholarship.org Powered by the California Digital Library
University of California

Game-Theoretic Safety Assurance for Human-Centered Robotic Systems

by

Jaime Fernández Fisac

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Engineering - Electrical Engineering and Computer Sciences

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor S. Shankar Sastry, Co-chair
Professor Claire J. Tomlin, Co-chair
Professor Anca D. Dragan
Professor Thomas L. Griffiths
Professor Ruzena Bajcsy

Fall 2019

Game-Theoretic Safety Assurance for Human-Centered Robotic Systems

Copyright 2019
by
Jaime Fernández Fisac

Abstract

Game-Theoretic Safety Assurance for Human-Centered Robotic Systems

by

Jaime Fernández Fisac

Doctor of Philosophy in Engineering - Electrical Engineering and Computer Sciences

University of California, Berkeley

Professor S. Shankar Sastry, Co-chair

Professor Claire J. Tomlin, Co-chair

In order for autonomous systems like robots, drones, and self-driving cars to be reliably introduced into our society, they must have the ability to actively account for safety during their operation. While safety analysis has traditionally been conducted offline for controlled environments like cages on factory floors, the much higher complexity of open, human-populated spaces like our homes, cities, and roads makes it unviable to rely on common design-time assumptions, since these may be violated once the system is deployed. Instead, the next generation of robotic technologies will need to reason about safety online, constructing high-confidence assurances informed by ongoing observations of the environment and other agents, in spite of models of them being necessarily fallible.

This dissertation aims to lay down the necessary foundations to enable autonomous systems to ensure their own safety in complex, changing, and uncertain environments, by explicitly reasoning about the gap between their models and the real world. It first introduces a suite of novel robust optimal control formulations and algorithmic tools that permit tractable safety analysis in time-varying, multi-agent systems, as well as safe real-time robotic navigation in partially unknown environments; these approaches are demonstrated on large-scale unmanned air traffic simulation and physical quadrotor platforms. After this, it draws on Bayesian machine learning methods to translate model-based guarantees into high-confidence assurances, monitoring the reliability of predictive models in light of changing evidence about the physical system and surrounding agents. This principle is first applied to a general safety framework allowing the use of learning-based control (e.g. reinforcement learning) for safety-critical robotic systems such as drones, and then combined with insights from cognitive science and dynamic game theory to enable safe human-centered navigation and interaction; these techniques are showcased on physical quadrotors—flying in unmodeled wind and among human pedestrians—and simulated highway driving. The dissertation ends with a discussion of challenges and opportunities ahead, including the bridging of safety analysis and reinforcement learning and the need to "close the loop" around learning and adaptation in order to deploy increasingly advanced autonomous systems with confidence.

[This page intentionally left blank]

To my parents, Concha and Curro, and to my sister Carmeluky.

¡Porque juntos somos geniales!

Contents

List of Figures

List of Tables

Acknowledgments

The time has finally come to put an end to a major phase of my life, one that arguably started in early 2013 with an admission email that kept me up all night in my bedroom in Madrid, my mind racing through the adventures that awaited me 10,000 kilometers away in Berkeley, California. Today, almost seven years later, I am sitting in the same room, looking back at the journey and the many travel companions without whose support and guidance it could not possibly have been completed. It is with deep gratitude that I dedicate these final words to them.

First and foremost, I am grateful to my three wonderful PhD advisors, Shankar, Claire, and Anca, for their unwavering support, wisdom, and kindness over the years. Any attempt to do them justice in writing here would be humorously inadequate, so I will instead say that I hope to be able to help others grow the way they have helped me.

I also owe much gratitude to the other two members of my dissertation committee: Ruzena, for encouraging me never to lose sight of the bigger picture, and for making all of us feel like a family; and Tom, for nurturing my fascination with human cognition and interaction. Both of them have actively shaped how I think about the relationship between technology and people.

The results and insights reached in my PhD work are the outcome of many close collaborations and stimulating discussions with my peers at Berkeley. Many thanks to my first research companions, Kene Akametalu, Jeremy Gillula, Shahab Kaynama, and Melanie Zeilinger, who made me feel welcome and valued from my early, inexperienced steps. Thanks to the fantastic *coffee spill gang*, Sylvia Herbert, David Fridovich-Keil, and Andrea Bajcsy, for all the late nights of coding and writing, endless Crazyflie experiments, and surreal acronym brainstorms; and for making sure I didn't accidentally skip meals at the peaks of research excitement. Thanks to Mo Chen and Somil Bansal for the discussions and joint discoveries around multi-agent systems, safety, real-world robotics, and how they all fit together; to Margaret Chapman for the thoughtful conversations about risk-sensitive safety; and to Jess Hamrick, Chang Liu, and Vael Gates for the interdisciplinary adventures bridging cognitive science and engineering. Thanks to Neil Lugovoy, Vicenç Rubies Royo, and Shromona Ghosh, for embarking together on a quest to tear down walls between research communities that should be speaking more to each other. Thanks also to Dexter Scobee and Andreea Bobu for their patience with my spotty presence during the last year and for making me feel like I never ceased to be a full-time member of the team. I am also grateful to my collaborators at the Center for Human-Compatible AI, who informed my thinking about our long-term responsibilities in the development of intelligent technologies. Finally, a special thank you to the younger students who put up with my far from infallible guidance: Ted Xiao, Elis Stefansson, Steven Wang, Eli Bronstein, Neil Lugovoy—I hope the experience of working together was as thrilling for them as it was for me.

I have been fortunate to have wonderful mentors during my PhD years: Sam Burden, Lillian Ratliff, Dan Calderone, Roy Dong, Aaron Bestick, Katie Driggs-Campbell, and Dorsa Sadigh have all held my hand more than once through the uncertainties of graduate school,

and have continued to do so from time to time after they graduated; and I am happy to still count them all as friends, and relieved to know they will pick up the phone anytime I need their guidance again.

Few things are as clear in my mind as the impossibility of having made it through graduate school without the superpowers of Jessica Gamble and Shirley Salanio, generously put to use in keeping students afloat in the sea of troubles, requirements, scheduling, travel, and paperwork that we sometimes feel lost in. Thanks for the countless times they have led my particular ship to safe harbor.

My excitement for the next steps is in conflict with the many great people that I know I will be missing when I move away—Joe Menke, Dapo Afolabi, David McPherson, Laura Hallock, Eric Mazumdar, Tyler Westenbroek, Ellis Ratner, Forrest Laine, Anusha Nagabandi, and the rest of the TRUST and SDH7 inhabitants: thanks to all of them for the many great discussions about research, life, the universe, and everything, the occasional early morning martial arts, and the never dull conference trips on both hemispheres.

A special mention needs to be made to my fellow *control freaks* who started the PhD journey with me and have accompanied me through all of its ups and downs: Cathy Wu, Roel Dobbe, Eric Kim, and Jason Poon. It has been a privilege learning and growing together over the last half-decade. May we always find the time to meet up for a late-night Top Dog. Thank you also to Jon Mather, Jess Lee, and Vinay Ramasesh for unquestionably making my time at Berkeley more eventful and enjoyable. The same is true of my fellow *Berkeley gang* Spaniards, Miquel Crusells Girona, Álvaro García-Delgado, Eduard Ansaldo Giné, Julia Gómez Camblor, Alejandro Castillejo Muñoz. . . a continually growing family that has helped me feel closer to home.

I spent my first year at Berkeley living in the International House, where I was fortunate to share a corridor and many fascinating impromptu discussions with Daniel Chada, who first piqued my interest in cognitive science research. My home for the next five years, a four-person apartment unofficially but solemnly called the *Batcave*, was also home, over the years, to Roel, Jon, Fanny, Linus, Ale, and Eduard, who I am happy to call my close friends and—just as important—my jamming companions. Their company always brought a pleasant closing note to long days in the lab.

Of all the people Berkeley has brought me close to, Tasneem is one I can never be grateful enough for. Since she entered my life a little over two years ago she has become essential to how I think about myself and the world. Her enthusiasm, generosity, and perseverance inspire me every day, and she can somehow make the most daunting challenges feel doable. I can only hope she will continue to light my way for many years to come.

The final thank you is to those who guided my steps before I was able to plan them on my own. To my many great teachers and early mentors. To my four loving grandparents. To Angelita, who always treated our family like her own. To my sister Carmen, who has long been my best mirror and knows the inside of my brain better than I do. And most of all, to our parents, who always encouraged us to follow our passion even if this would lead us far away from home. Their selfless love is ultimately the reason this dissertation exists.

Chapter 1

Introduction

> The danger which is least
> expected soonest comes to us.
>
> Voltaire (1694–1778)

It is a time of rapid change in humanity's technological development. Automation technologies, spurred by the increasing availability of computer hardware, high-bandwidth telecommunications, and substantial advances in modern artificial intelligence, are becoming increasingly pervasive in human society. Today, the domain of automation transcends industrial manufacturing, and permeates human activities and infrastructure: homes and workplaces are becoming widely retrofitted with "smart" appliances, and our utilities, from electricity to internet connectivity, are supplied through complex self-regulation schemes. The rapidly growing "tech" industry continues to expand the capabilities of personal computers and mobile devices, which increasingly mediate our interaction with the world and with each other through a plethora of software "apps", providing us with a relentless stream of automatically generated information, recommendations, and services.

A central front in this progress of automation technology is the advancement of robotics. Once relegated to factory floors, robotic systems—mechanical devices with the ability to sense their environment, make decisions, and perform actions that modify it—are beginning to populate the human space. Robotic vacuum cleaners are found in many homes [1]; surgical robots enable surgeons to perform high-precision operations with minimal invasiveness and patient recovery time [2]; commercial drones for personal and professional use are sold by tens of thousands of units worldwide every month, for purposes ranging from photography and film to construction, agriculture, and even newly emerging *drone racing* competitions [3]; and crucially, autonomous driving is expected to drastically reduce the rate of accidents and fatalities on the road, while transforming the morphology of urban spaces by inducing a widespread shift towards a *sharing economy* of transportation [4]. These new technologies are also having a direct impact in the developing world: lightweight unmanned aircraft are already being used to deliver medicine and blood supplies to hospitals at locations that are

hard (and occasionally impossible) to reach by road, requiring minimal human supervision during their flight [5].

It seems clear that robotic technologies present a vast range of opportunities for empowering human activities and improving our individual and collective welfare. At the same time, the deployment of these complex automated systems brings important new questions around safety and reliability. If an autonomous car takes inappropriate actions, it may cause injury or death not only to its own occupants, but to other road users around it. Many of the robotic systems that we expect to develop and deploy in the coming years and decades are *safety-critical*, that is, their incorrect operation could lead to severe failures that we should strive to avoid at all costs.

The focus of this dissertation is on how to formally reason about the safe operation of current and future robotic systems, how—and to what extent—we may provide meaningful assurances around these technologies, and how these assurances can be actively represented, and reasoned about, by the systems' own automated decision-making processes. We will pay special attention to the challenges arising from the complex, large-scale interactions involved in multi-agent systems such as unmanned aircraft system (UAS) traffic, the increasing role of learning-based and data-driven algorithms in the control of robotic systems, and the need for these systems to coexist and interact with human beings whose behavior may be extremely difficult to predict accurately. In spite of these challenges, we will show that there are tractable, rigorous, and effective methods that can enable robotic systems to preserve safety with high confidence even in the presence of substantial uncertainty about the world.

1.1 Central Challenges in Robotic Safety Assurance

1.1.1 Multi-Agent Systems

As robotic systems step out of their traditional cages on factory floors and into open environments, they cannot be expected to operate in isolation. Many environments where we expect to deploy—and are indeed beginning to deploy—modern robotic systems are populated by humans in different capacities. One of the central challenges faced by the nascent autonomous driving industry is designing automatic decision-making schemes that can account not only for the vehicle's physical evolution, but for the active, often strategic responses of other road users and how these will condition future decisions in turn. This interaction makes the problem of safe driving fundamentally game-theoretic, requiring a depth of analysis that has traditionally been beyond the scope of robotic motion planning and control.

On the other hand, robotic systems will often not be deployed one at a time, but forming multi-robot teams or networks in which communication and cooperation are not only feasible, but possibly necessary for successful and safe operation. In the United States, NASA has been conducting active research over the last decade on a new, highly-automated air traffic control paradigm for unmanned vehicles, meant to manage routing and ensure separation for vehicle densities far exceeding the capacity of human air traffic controllers [6]; similar

initiatives are underway in the European Union. Even with the assumption of coordination, the planning space is intrinsically combinatorial, meaning that the necessary computation to reason about safety and efficiency tends to increase exponentially with the number of vehicles. Finding scalable schemes to compute safe, conflict-free trajectories for large numbers of vehicles is crucial to the successful deployment of unmanned aircraft services in the coming years.

1.1.2 Learning-Based Systems

Much of the recent progress in modern artificial intelligence is linked to advances in machine learning, which enables software systems to make decisions based on statistical inferences from data. In particular, machine learning techniques have led to considerable advances in the processing of high volumes of sensory information, in contexts such as speech recognition and computer vision, the latter of which is particularly central to robotic perception. In addition, reinforcement learning provides a data-driven approach to sequential decision problems such as robotic planning and control. These techniques have seen significant improvement in roughly the last half-decade through the development of hyperparametric function approximators commonly known as deep neural networks [7, 8].

The state-of-the-art performance delivered by machine-learning methods in a number of relevant computational problems, as well as their ability to adapt to observed data, motivates their use in automation technologies and robotic systems in particular. Unfortunately, in contrast with traditional logic-based software, the behavior of learning-based software cannot be determined by examining the program's source code, but depends heavily on the data that the software has been *trained* on. As a result, formal correctness verification of machine-learning systems is extremely challenging in some cases and infeasible in others. This is especially true of "black-box" systems such as neural networks, where it has not generally been possible to determine reliably what it is that the system has learned,[1] let alone give any *a priori* guarantees about what the system will learn once exposed to certain data. Indeed, one notorious weakness of many machine-learning methods is their brittleness to "out of distribution" data, which makes them highly vulnerable to adversarial attacks and unreliable for deployment in uncertain environments [10].

This lack of functional guarantees has largely prevented the application of learning-based methods to safety-critical or high-stakes systems, while a number of contexts where they have been applied have seen serious issues related to unexpected and poorly understood behavior (from chat bots using offensive and antisocial language to decision-making systems discriminating against individuals on the basis of race or socioeconomic status). How can we integrate learning-based components into automation systems and robotic platforms in a way that allows us to reap the benefits in terms of performance and adaptability while retaining the ability to provide meaningful assurances about the overall operation?

[1] Although recent efforts have had some success in providing input-output bounds on trained neural networks [9].

A recurring theme throughout this thesis will be the observation that safety and learning need not be at odds with each other, and can in fact function in a synergistic and complementary fashion in an automation system.

1.1.3 Human-Centered Systems

To the extent that robotic systems will be deployed in environments shared with human beings, their decisions will often need to be informed by predictions of human behavior. Safety, in particular, may depend not only on the actions taken by the robotic system, but on those taken by humans around it. How can a system provide safety assurances in the presence of uncontrolled human agents?

On the one hand, the worst-case analysis techniques that often prove valuable in computing robust guarantees under physical uncertainty quickly become impractically conservative in the case of uncertain human behavior: if the robot considers all actions that a human could *physically* take, it may conclude that it is impossible to remain safe, even in standard conditions of operation. Therefore, it is often necessary to reason about what human actions are *likely* in a given situation, typically through cognitive models of human decision-making.

On the other hand, given the complexity of human behavior, any predictive model is bound to be incomplete and therefore inaccurate in certain conditions; in such situations, human actions that had been deemed unlikely by the robot's model may in fact compromise safety when they do take place. In scenarios like autonomous driving or physical human-robot interaction, it is critical that robotic systems respond promptly and gracefully when the accuracy of their predictions degrades.

Finally, the future behavior of people interacting with a robotic system is often heavily dependent on the system's own upcoming actions. This links directly to the multi-agent aspect of robotic system operation: failing to account for the mutual, often strategic, interplay between the decisions of human beings and robotic systems can lead to incorrect robotic decisions that potentially result in loss of safety.

Many important lessons about the criticality of human-automation interaction in system safety can be drawn from the aerospace sector, from the early discovery of "pilot-induced oscillations", resulting from the delayed feedback loop between the pilot's corrections and the aircraft's response, to the recent accidents involving the Boeing 737 MAX, where a malfunctioning automated "safety" override meant to correct pilot inputs unexpectedly placed the aircraft in a fatal nose dive [11, 12]. Even in an industry that enjoys a well-deserved safety reputation, incidents of this kind have kept resurfacing in different forms over the decades, showing that incorrect design assumptions about the interaction between the human crew and the on-board automation can have catastrophic consequences.

Meanwhile, on the ground, autonomous driving technology is similarly finding human behavior modeling to be a central challenge. Here, the difficulty in modeling human decisions compounds with the challenge of accurately reasoning about strategic road interactions. Improving the prediction and planning algorithms in the autonomy stack to more competently

handle uncertain interactions with real, model-defying human road users is currently a key open problem under study by researchers and practitioners in the sector.

The design of safe human-centered robotic systems therefore requires a careful analysis of the interaction between the automated components and human actors, while at the same time making the overall operation as robust as possible to misspecified models and inaccurate assumptions about human behavior.

1.2 Thesis Overview and Contributions

The central contention of this Ph.D. dissertation is that ensuring safe robotic operation under an inevitably incomplete understanding of the world and other agents requires bridging *model-based* and *data-driven* formulations. Autonomous systems will need to compute theoretically sound guarantees grounded in available knowledge and data, but also respond resiliently to unexpected environment changes that could invalidate the computed guarantees.

Our approach to safety assurance will therefore combine robust optimal control theory with Bayesian machine learning, incorporating insight from cognitive science and dynamic game theory to understand closed-loop interaction with human beings. Ultimately, providing reliable assurances requires robotic systems to reason explicitly about the reality gap between their models and the real world, from unforeseen physical disturbances to unexplained human actions, and decide which guarantees to rely on in light of current observations.

The work in this thesis complements basic theoretical groundwork with extensive testing on physical robotic platforms and studies with human participants. The organization and key contributions are summarized below, and can be visualized in Figure 1.1.

First, Chapter 2 covers some important concepts that will serve as the theoretical foundation for the contributions made in this thesis. In particular, the chapter introduces the necessary background in dynamical systems theory, optimal control, game theory, machine learning and human decision modeling, and establishes a consistent technical notation that will be used throughout the thesis.

Part I: Safety Analysis for Robotic Systems. The first part of the thesis presents core theoretical approaches and algorithmic machinery used for sound and scalable safety analysis under uncertainty. Chapter 3 begins with the formulation of *time-varying* Hamilton-Jacobi analysis for reach-avoid differential games and introduces the novel *double obstacle Isaacs equation* which is a generalization of the existing time-invariant formulation. Crucially, the viscosity solution to the double-obstacle equation—proven to be the value function of time-varying game—can be obtained in roughly the same amount of computation as in time-invariant settings. This theoretical result is then applied in Chapter 4 to obtain a scalable scheme for planning and execution of safe trajectories for cooperative multi-robot systems. The approach is demonstrated in the context of large-scale unmanned air traffic management, and made robust to modeling error through further Hamilton-Jacobi analysis. Chapter 5 addresses safe trajectory planning in real time by combining simplified dynamical

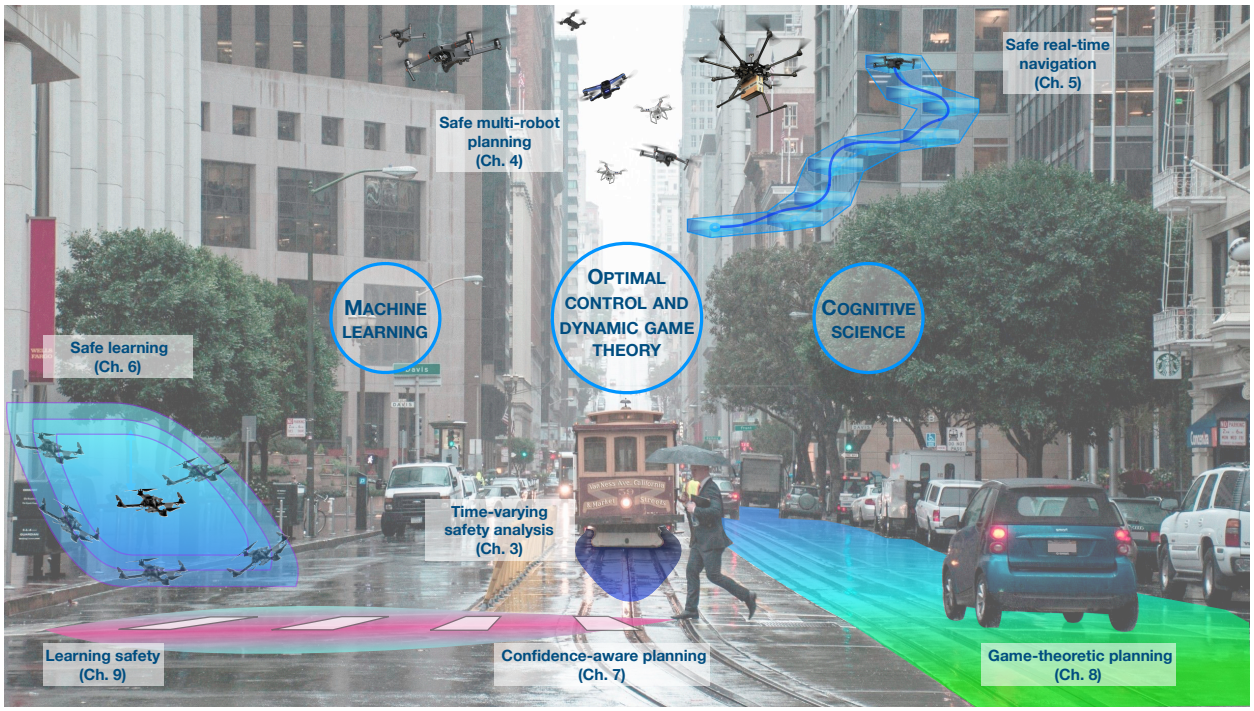


Figure 1.1: Thesis overview illustrated on an urban scene.

models for fast online computation with high-fidelity representations for robust pre-computed tracking guarantees. The resulting robust splanning scheme is additionally endowed with recursive feasibility under imperfect environment information and finite planning horizons. The results in this part of the thesis were originally introduced in [13–17].

Part II: Safety Across the Reality Gap. The second part of the thesis engages with the central engineering challenge of providing safety assurances for real systems in spite of the inevitable inaccuracies of any theoretical model. These chapters look into the ties between real-time assurances and online learning and explore the important special case of safe robotic decision-making in the midst of human beings. Chapter 6 applies the Hamilton-Jacobi safety machinery to construct a general *safe learning* framework enabling robotic systems to execute learning-based control schemes such as reinforcement learning. The framework additionally leverages online data to refine the safety analysis over time and continually validate the applicability of theoretical safety guarantees in light of the observed divergence between modeled and real system behavior, thereby closing a synergistic loop between safety and learning. Chapter 7 introduces a real-time Bayesian confidence inference scheme for robotic systems to continually monitor the accuracy of their predictive human models as they navigate a shared environment with people. Whenever the observed human actions are not well-explained by the predictive model, predictions of future motion automatically become more uncertain, causing the robot to increase caution until model accuracy increases. Chapter 8 then considers the coupling between the prediction and planning problems when

a robotic system is interacting with a human in a strategic context, specifically in a driving scenario. Here we apply a full game-theoretic treatment, and introduce a hierarchical scheme that enables the use of high-level strategic reasoning over time horizons of a few seconds in conjunction with low-level tactical trajectory planning over shorter horizons. The results in this part of the thesis were originally introduced in [18–21].

Part III: Safe Steps Forward. The final part of the thesis discusses some of the key technical challenges lying ahead on the road towards safe advanced autonomous systems. Chapter 9 draws a promising connection between safety analysis and modern artificial intelligence techniques, rendering Hamilton-Jacobi safety analysis compatible with the reinforcement learning formulation through a time-discounted modification that induces a required contraction mapping. This opens the door to the application of the growing plethora of deep reinforcement learning schemes to safety analysis, enabling the computation of efficient best-effort safety-preserving control policies for previously intractable high-dimensional systems. More broadly, the new formulation facilitates incorporating safety requirements into reinforcement learning systems beyond robotics. Finally, Chapter 10 presents some concluding remarks and offers a look at the future of autonomous safety assurance, not only in robotic systems but in the wider family of intelligent technologies that are increasingly pervading our society.

Chapter 2

Background and Preliminaries

> All models are wrong, but some are useful.
>
> George Box (1919–2013)
> Statistician

2.1 System Dynamics and Model Uncertainty

Through much of this dissertation we will be concerned with the operation and evolution of physical systems. The class of systems that we will consider will be quite broad, encompassing a variety of robotic platforms, from drones to autonomous cars, and in some cases also including human beings. In its most general sense, a *system* may be any part of the world whose internal functioning and external interactions are of relevance to us. This general definition allows us to talk equally meaningfully about a robotic system, a socio-political system, or a planetary system. The study of any such systems is often involved with how they function, what are the principles that govern their behavior, and hence what their evolution over time may be. A unifying framework under which these questions can be studied is the mathematical theory of *dynamical systems*. This formal machinery allows us to describe the evolution of the system through the notion of its *state*, a minimal collection of variables that suffices to characterize and predict its future behavior.

In the context of robotics, control theory, and artificial intelligence, we are typically concerned with systems whose evolution over time is at least in part a function of certain physical variables whose values or behaviors we can choose, usually termed *control inputs*. For example, the evolution of a car driving on a road is affected by the motion of the steering wheel, transmitted to the steering column and ultimately the wheels; if we are partially or fully automating the behavior of the car, then it is useful to model the steering wheel angle (or more often its time derivative) as a control input, that is, a variable that we can arbitrarily set at any given moment.

In addition, since what is ultimately of interest is the evolution of the real system rather than our abstract mathematical representation of it, it is often important to explicitly account for uncertainty in the system model. A useful abstraction that can represent this uncertainty in a general way is the introduction of a *disturbance input*, analogous to the control input with the important difference that we are not able to choose, or even accurately predict, the values that it will take over time.

The analysis of how a dynamical system evolves can be done in continuous or discrete time, and the uncertainty can be treated under different modeling frameworks. We will dedicate this section to an overview of these models, which will serve as a common foundation for much of the mathematical machinery introduced in this chapter.

A note on terminology. The study of dynamical systems, and how to steer their evolution towards desirable outcomes, has been undertaken by multiple fields of mathematics, science, and engineering, often arriving at similar insights from different perspectives. These disciplines include dynamical systems theory, control theory, optimization theory, probability theory, decision theory, and game theory. The resulting theories and analytical tools have found use in application domains as diverse as robotics, machine learning, econometrics, cognitive science, and biology, among others. This variety of perspectives highlights the central importance of the dynamic decision-making problem; it also introduces some amount of crosstalk in the literature at large. It is not uncommon to find different terminology applied to equivalent concepts: for example, the *inputs* in control theory are often referred to as *actions* in decision theory, and *control laws* in the former become *policies* in the latter. Table 2.1 presents a summary of some relevant concepts, as denoted in different fields.

Control theory	Artificial Intelligence	Game theory
System	Environment	Game
Controller	Agent	Player
Dynamics	Transition	Dynamics
State	State	State
Control input	Action	Play
Control law	Policy	Strategy
Cost (minimize)	Reward (maximize)	Payoff (maximize)

Table 2.1: Analogous terms between control theory, artificial intelligence, and game theory.

While this lexical divide may be inevitable to some extent, it is far from insurmountable. Indeed, some of the key contributions in this work are built on the rapprochement between different but closely related formulations such as optimal control and reinforcement learning. Where possible, this dissertation will seek to use wording that can, on the one hand, be easily understood by readers familiar with any one of these disciplines and, on the other, remind us of how closely related these areas of study truly are.

2.1.1 Continuous-Time System Dynamics

Many physical systems can be modeled through differential equations that describe the underlying phenomena (mechanical, electrical, etc.) governing their evolution. Models derived from first principles such as Newton's second law have been successfully used to predict and control the behavior of a wide variety of physical systems.

Consider the dynamical system's continuous state $x \in \mathbb{R}^n$, whose evolution may be influenced by one or more agents.[1] In the simplest case, we may have a single agent, namely the automated controller we are interested in designing. We can write the evolution of the system through the dynamics

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

where $u \in \mathcal{U} \subseteq \mathbb{R}^{n_u}$ denotes the *control input* to the system. We will assume the set $\mathcal{U}$ to be compact, since the amount of physical control effort that our controllers can exert (force, torque, voltage, etc.) is always bounded in practice. While the dynamics f will often be time-invariant, allowing us to drop dependence on t, this is not always the case, and we will explicitly include time dependence where relevant (it will play a particularly important role in Chapter 4).

In other cases we may have multiple agents simultaneously making decisions that affect the evolution of the system. These agents may be multiple automated controllers operating in a decentralized or distributed fashion (as will be the case when we study multi-vehicle trajectory planning for large-scale unmanned air traffic in Chapter 4), and we may be interested in designing the decision rules for all of them. On the other hand, some of the agents may implement decision rules that we are not able to choose or exactly predict at design time. An important class of such agents are human actors, and human-automation systems will be the focus of Chapters 7 and 8.

Assigning each agent a unique index[2] $i \in \mathcal{I}$, and denoting their respective control inputs $u_i \in \mathcal{U}_i \subseteq \mathbb{R}^{n_{u_i}}$, with $\mathcal{U}_i$ assumed compact, the evolution of the state can be expressed through the dynamics

$$\dot{x} = f\left(x, (u_i)_{i \in \mathcal{I}}, t\right) \ . \tag{2.2}$$

Finally, in most practical cases we cannot assume to know the dynamics exactly, and the actual evolution of the system may depend on the realization of some uncertainty about the world. It is often useful to explicitly formalize the effect of this uncertainty on the dynamics, typically by admitting an additional input source, which we will term the *disturbance* $d \in \mathcal{D} \subseteq \mathbb{R}^{n_d}$. This uncontrolled and *a priori* unknown input can be used to model actual

[1] For the most part, we will assume that the state space is $\mathbb{R}^n$, while noting that in some relevant cases we may have different state spaces *embedded* in $\mathbb{R}^n$ (for instance, the angular pose of a robotic joint or rigid body in 2- or 3-dimensional space evolves in the orthogonal groups $SO(2)$ and $SO(3)$ respectively). In this dissertation we will mostly eschew the more involved mathematical treatment of differential equations on manifolds, and direct the interested reader to [22], Ch. 8, for an in-depth exposition.

[2] It may be convenient to think of the index i as simply taking values $1, 2, \ldots, N$. More generally, we can define arbitrary indices such as H for the human agent and A for the autonomous system. We therefore keep the notation general using index set $\mathcal{I}$.

physical disturbances that we may expect our system to encounter in the environment, such as wind gusts perturbing the flight of an aerial vehicle. Its use, however, is much more general: it can also be used to account for the various discrepancies that might occur between the nominal evolution predicted by the model and the actual observed behavior of the physical system, whether these are due to unmodeled higher-order dynamics, unexpected environmental conditions, or unknown agents taking purposeful actions in pursuit of their own goals. We will assume that $\mathcal{D}$ is a compact set, which is not a stringent condition in itself, since it amounts to requiring that all disturbances are finite in magnitude[3]. The *uncertain* dynamics can then be expressed as

$$\dot{x} = f\left(x, (u_i)_{i\in\mathcal{I}}, d, t\right) \quad . \tag{2.3}$$

If nothing is said about the values taken by d, we are left with an undefined evolution of our system, even when the state x and every agent's input u_i are known. However, once we properly quantify our uncertainty over d, this will translate into a well-defined uncertainty about the evolution of the system. We will see that there are two important formalisms under which this uncertainty can be characterized, corresponding to two schools of thought in decision-making theory: probabilistic and robust analysis.

Before that, we need to introduce some technical conditions for continuous-time state trajectories to be uniquely defined. The flow field $f : \mathbb{R}^n \times \prod_{i\in\mathcal{I}} \mathcal{U}_i \times \mathcal{D} \times \mathbb{R} \to \mathbb{R}^n$ is assumed uniformly continuous and bounded, as well as Lipschitz in x at all times t and for all input values $(u_i)_{i\in\mathcal{I}}$ and d.

Letting $\mathbb{U}_{i,t_0}$ and $\mathbb{D}_{t_0}$ denote the collections of measurable[4] functions $\mathbf{u}_i : [t_0, \infty) \to \mathcal{U}_i$ (for $i \in \mathcal{I}$) and $\mathbf{d} : [t_0, \infty) \to \mathcal{D}$ respectively, and allowing the agents and disturbance to choose any such signals, the evolution of the system from any initial state x is determined (see for example [24], Ch. 2, Theorems 1.1, 2.1) by the unique continuous trajectory $\mathbf{x} : [t_0, \infty) \to \mathbb{R}^n$ solving

$$\begin{aligned} \dot{\mathbf{x}}(t) &= f\big(\mathbf{x}(t), (\mathbf{u}_i(t))_{i\in\mathcal{I}}, \mathbf{d}(t), t\big), \text{ a.e. } t \geq 0 \ , \\ \mathbf{x}(t_0) &= x_0 \ . \end{aligned} \tag{2.4}$$

[3]The finite magnitude of disturbances does imply that the physical state of the system is not allowed to instantaneously jump between values, that is, it involves an epistemic commitment to the continuity of trajectories. For some systems, it is in fact convenient to model certain events in the evolution of the state as though discontinuities in the state trajectory do take place: a notable example is legged locomotion, where the contact between a leg and the terrain is often modeled through an instantaneous change in velocity. While this is admittedly a simplification of the complex contact dynamics that take place, in such cases it is beneficial to relax the continuity condition. There has been a rich theoretical development of the theory of such *hybrid systems*, which combine continuous *flow* dynamics with discrete *jumps*. The theory presented here is consistent with the *flow* part and can in fact be used within the more general hybrid system framework (see [23]). To avoid introducing a heavier mathematical scaffolding than necessary, we will mostly be obviating the hybrid aspect in this dissertation.

[4]A function $h : X \to Y$ between two measurable spaces (X, Σ_X) and (Y, Σ_Y) is said to be measurable if the preimage of a measurable set in Y is a measurable set in X, that is: $\forall V \in \Sigma_Y, h^{-1}(V) \in \Sigma_X$, with Σ_X, Σ_Y σ-algebras on X,Y.

This is a solution in what is known as Carathéodory's *extended sense*, that is, it satisfies the differential equation at *almost every instant* (i.e. except on a subset of Lebesgue measure zero). It is a relatively general solution concept that will allow us to use different uncertainty models throughout this dissertation while ensuring that trajectories are mathematically well-defined.

In certain cases, we may only consider the evolution of the system on a compact time interval $[t_0, t_f]$, in which case we will let $\mathbb{U}_{i,t_0}^{t_f}$ and $\mathbb{D}_{t_0}^{t_f}$ denote the collection of measurable control signals $\mathbf{u}_i : [t_0, t_f] \to \mathcal{U}_i$ and $\mathbf{d} : [t_0, t_f] \to \mathcal{D}$ respectively.

Following the standard notation in dynamical systems theory, let $\mathbf{x}(\,\cdot\,; x_0, t_0, (\mathbf{u}_i)_{i\in\mathcal{I}}, \mathbf{d})$ denote the state trajectory starting at state x_0 at time t_0 and steered by control inputs $(\mathbf{u}_i)_{i\in\mathcal{I}}$ and disturbance input $\mathbf{d}$ over time. For compactness, we will frequently use the alternative shorthand notation $\mathbf{x}_{x_0,t_0}^{(\mathbf{u}_i),\mathbf{d}}(\cdot)$ to refer to the same trajectory.

Feedback Control Policies

The above trajectory definition requires specifying the control signals that agents will input over time. However, since the evolution of the state is uncertain, it is rarely appropriate for agents to commit to a control signal and continue to execute it without accounting for how the system is actually evolving. Such a control scheme is known as *open-loop*, in reference to the perception-action *feedback loop* not being "closed", since information about the state is not informing the control inputs.

Feedback is historically and fundamentally at the heart of automation (cf. [25]), and its ubiquitous use is motivated by the practical impossibility of exactly predicting how our control inputs will cause a system to evolve. Consistently, we will usually allow the control signals to adapt at each moment in time to the current state of the system. This gives rise to the notion of a *feedback control policy* (or *feedback strategy* in dynamic game theory), namely a function $\pi_i : \mathbb{R}^n \times \mathbb{R} \to \mathcal{U}_i$ that maps the current state of the system (and possibly the current time[5]) to a control input $u_i = \pi_i(x, t)$.

This information structure is known as *perfect-state closed-loop feedback*, or simply *state feedback* or *feedback*. It implies the assumption that agents have the ability to accurately and instantaneously perceive the current state of the system. This is not a trivial approximation in general—although it is one that we will be making through most of this dissertation—and we will revisit its implications towards the end of the section.

A theoretical issue that arises when introducing feedback is whether trajectories continue to be well defined under arbitrary control policies. The answer is in fact not always in the affirmative, and pathological cases exist where even seemingly innocuous feedback control policies may render system trajectories ill-defined, as illustrated by the following example.

[5]While time-invariant control policies are common, and we will drop time dependence in such cases, time-varying policies can be appropriate even in time-invariant settings, particularly under finite time horizons.

Example 2.1. *Consider a simple* 1*–dimensional dynamical system with state* $x \in \mathbb{R}$ *and control input* $u \in [-1, 1]$*, whose dynamics are given by*

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

and choose the time-invariant feedback control policy $\pi : \mathbb{R} \to [-1, 1]$ *defined as*

$$\pi(x) = \begin{cases} 1 & x < 0 \\ -1 & x \geq 0 \end{cases}$$

Even though f *is uniformly continuous, Lipschitz (constant) in* x*, and bounded, no solutions exist starting at, or passing through,* $x = 0$*, even "at almost all times" as required by Carathéodory's extended solution concept. Indeed, if a trajectory finds itself at exactly* $x = 0$*, its time derivative must be* $\dot{x} = -1$*, which would immediately take it into the negative half-space, requiring its derivative to be* $\dot{x} = 1$*, and propelling it back to the origin, where the time derivative would be reversed once again. This would happen infinitely many times on any bounded time interval, something termed "Zeno behavior" after the Greek philosopher Zeno of Elea (5th century B.C.), who proposed multiple paradoxes related to infinitely many events taking place in finite time.*[a] *Zeno behavior is a well-known problematic artifact emerging from overly simplified abstraction of system dynamics [26]. While no physical system is Zeno (with any "instantaneous"switches actually involving unmodeled dynamics, e.g. voltage changes), Zeno behavior in a dynamical model can impede accurate analysis of the system of interest.*

[a] The reader may be familiar with the paradox of *Achilles and the tortoise*. The fleet-footed Achilles races against a much slower tortoise, giving it a head start. He presently covers the initial separation, but by then the tortoise has made additional headway; Achilles quickly covers that new distance only to find that the tortoise is ahead once again; and so on, argues Zeno, until Achilles gives up and accepts his defeat.

To avoid such pathological cases, we would generally like to disallow control policies that result in non-measurable input signals over time. Note that a sufficient condition is for π_i to be Lipschitz in x and measurable in t, so that the resulting closed-loop dynamics $f_{(\pi_i)}(x, d, t) := f\big(x, (\pi_i(x, t))_{i\in\mathcal{I}}, d, t\big)$ continue to be Lipschitz in x and measurable in t. However, this is more restrictive than necessary, since discontinuities are possible in many cases. In the above example, if we minimally modify the control policy so that $\pi(0) = 0$ it is easy to check that Carathéodory solutions are globally well defined as $\mathbf{x}_{x_0,t_0} = \operatorname{sgn}(x_0) \max\{0, |x_0| - (t - t_0)\}$. Later in this chapter, we will see that continuous-time optimal control and safety analysis can be formulated in the space of measurable signals, implicitly defining feedback policies that, by construction, result in well-defined system trajectories.

Slightly overloading the previous trajectory notation, we will let $\mathbf{x}(\,\cdot\,; x_0, t_0, (\pi_i)_{i\in\mathcal{I}}, \mathbf{d})$

denote the state trajectory starting at state x_0 at time t_0 and steered by feedback control inputs $\mathbf{u}_i(t) = \pi_i\big(\mathbf{x}(t)\big)$ and disturbance input $\mathbf{d}$ over time. Equivalently, we will use the compact notation $\mathbf{x}_{x_0,t_0}^{(\pi_i),\mathbf{d}}(\cdot)$.

2.1.2 Discrete-Time System Dynamics

In some cases it will be convenient to consider the evolution of the system by quantifying changes over discrete time steps. These time steps $k = 0, 1, 2, \ldots$ are often chosen to be spaced at regular intervals $\delta > 0$, so that $t^{k+1} - t^k = \delta$ for all k. We can then express the dynamics of the system using a finite difference equation

$$x^{k+1} = f^k\left(x^k, (u_i^k)_{i\in\mathcal{I}}, d^k\right) \ , \quad k = 0, 1, 2, \ldots \tag{2.5}$$

This alternative representation can be seen as a convenient simplification of the continuous-time analysis that enables us to work with sequences of states and inputs rather than continuous trajectories and signals as functions of time. It is not meant to imply that the system *actually* evolves in discrete increments, and proper use of this representation, e.g. in robot motion planning, is accompanied by an analysis of what may happen *in between* time steps, e.g. whether collisions with obstacles may take place.

These dynamical representations constitute useful models that allow us to make informed predictions and decisions. At the same time, to the extent that all models have limited accuracy, quantifying their uncertainty is critical to making meaningful statements and assurances about the real system's behavior (and not only about the idealized behavior of the mathematical abstraction that is the model).

2.1.3 Robust Analysis: Dynamic Inclusion

One of the two central approaches for formalizing and working with uncertainty is robust analysis. In this framework, one specifies the *set* of possible realizations of the system's evolution, or equivalently, the set $\mathcal{D}$ of possible disturbances d that may affect the system's dynamics. This leads to the notion of a *dynamic inclusion*. Rather than a dynamic equation, in which we specify exactly what the state derivative $\dot{x}$ or the next state x^{k+1} will be, we instead allow a set of possible evolutions through a differential inclusion

$$\dot{x} \in F(x, (u_i)_{i\in\mathcal{I}}, t) := \{f\left(x, (u_i)_{i\in\mathcal{I}}, d, t\right),\ d \in \mathcal{D}\} \ , \tag{2.6}$$

or, in the discrete-time formulation,

$$x^{k+1} \in F(x^k, (u_i^k)_{i\in\mathcal{I}}, t^k) := \left\{f\left(x^k, (u_i^k)_{i\in\mathcal{I}}, d\right),\ d \in \mathcal{D}\right\} \ . \tag{2.7}$$

In the discrete-time case, the inclusion $F(x^k, (u_i^k)_{i\in\mathcal{I}}, t^k)$ is also called the one-step *forward-reachable set* from x^k under control inputs $(u_i^k)_{i\in\mathcal{I}}$. Whether in discrete or continuous time, the robust formulation allows us to reason about what states may be reached from a given

initial configuration (*forward-reachable set*), as well as what states may evolve into a certain terminal configuration (*backward-reachable set*). We will view this in more detail in Section 2.3.

The design of our autonomous controller should then ensure that the resulting system behavior is appropriate for *all* possible realizations of the uncertainty. In particular, we will want to guarantee that certain properties will hold even for the *worst* possible realization of the disturbance. As we will see in Section 2.3, if a feedback controller can be guaranteed to enforce a property under the worst-case disturbance, then by construction this controller will also successfully enforce the desired property under all other realizations of the disturbance. This worst-case analysis, sometimes referred to as *tychastic* analysis in contrast to stochastic analysis [27], allows formulating strong guarantees about the system's performance and safety for *all* conditions within a certain class.

In some cases it will be useful to consider disturbance-affine systems, that is, systems in which the disturbance enters the system dynamics additively. In these cases the disturbance can be thought of as directly encoding the discrepancy between the nominal behavior of the model and the one followed by the physical system. The safe learning framework presented in Chapter 6 will formalize the uncertain system dynamics in the form

$$\dot{x} = f\left(x, (u_i)_{i\in\mathcal{I}}, d\right) = f_x\left(x, (u_i)_{i\in\mathcal{I}}\right) + f_d\left(d\right) \quad , \tag{2.8}$$

with f_x and f_d satisfying the same boundedness and continuity conditions as f, and f_d additionally being injective onto its image (to prevent overparametrization of model-system discrepancy). As we will see, it will be useful to think in terms of a state-dependent uncertainty, since we may have more confidence in the accuracy of our model under some operating conditions than we do under others. We will, in those cases, use a state-dependent uncertainty set through a set-valued map $\hat{\mathcal{D}} : \mathbb{R}^n \to 2^{\mathcal{D}}$ assigning to each state x a corresponding uncertainty set $\hat{\mathcal{D}}(x) \subseteq \mathcal{D}$, under mild technical conditions that ensure that system trajectories remain well-defined.

2.1.4 Probabilistic Analysis: Transition Measure

The other central approach to decision-making under uncertainty is probabilistic analysis. Rather than only specifying what evolutions of the system are possible, this framework further quantifies *how likely* different evolutions are, which can be formalized as drawing the disturbance $d \in \mathcal{D}$ from a probability distribution, that is, a measure P_d over $\mathcal{D}$ such that $P_d(\mathcal{D}) = 1$. Therefore, the evolution of the system is modeled as a *stochastic process*.

In continuous time, certain characterizations of the disturbance's distribution lead to a well-defined stochastic differential equation. In this dissertation, we will eschew the mathematically cumbersome stochastic differential equation formulation, and instead focus our attention on discrete-time stochastic dynamics, induced by drawing d in (2.5) from a well-defined probability distribution P_d over $\mathcal{D}$. This results in a Markov decision process (MDP) in the one-agent case and a (Markov) stochastic game in the multi-agent case. The discrete-time stochastic dynamics are sometimes expressed through a *transition measure* P_x over $\mathbb{R}^n$,

giving the probability of the next state x^{k+1} conditioned on the current state x^k and control inputs $(u_i^k)_{i\in\mathcal{I}}$. For any measurable $\mathcal{X}^{k+1} \subseteq \mathbb{R}^n$:

$$P_x\left(\mathcal{X}^{k+1} \mid x^k, (u_i^k)_{i\in\mathcal{I}}\right) = \int_{\mathcal{D}} \left[f^k\left(x^k, (u_i^k)_{i\in\mathcal{I}}, d\right) \in \mathcal{X}^{k+1}\right] dP_d(d) \ , \tag{2.9}$$

with $[\cdot]$ the Iverson bracket, which evaluates to 1 if the proposition inside is true and 0 if it is false.

This Markovian model allows reasoning about state probabilities at arbitrary times in the future. In particular, for any number of steps l into the future, letting $\mathbf{u}_i := (u_i^k, \ldots, u_i^{k+l-1})$, we have

$$\begin{aligned} P_x\left(\mathcal{X}^{k+l} \mid x^k, (\mathbf{u}_i)_{i\in\mathcal{I}}\right) = \int_{\mathbb{R}^{n^{k+l-1}}} \cdots \int_{\mathbb{R}^{n^{k+1}}} & P_x\left(\mathcal{X}^{k+l} \mid x^{k+l-1}, (u_i^{k+l-1})_{i\in\mathcal{I}}\right) \\ & dP_x\left(x^{k+l-1} \mid x^{k+l-2}, (u_i^{k+l-2})_{i\in\mathcal{I}}\right) \cdots \\ & dP_x\left(x^{k+1} \mid x^k, (u_i^k)_{i\in\mathcal{I}}\right) \ . \end{aligned} \tag{2.10}$$

Rather than completely marginalizing out the intermediate states, we may want to reason about the probability measure of a particular trajectory "tube" (that is, the Cartesian product of subsets $\mathcal{X}^k$ over time). This can be done by changing the integration limits accordingly:

$$\begin{aligned} P_{\mathbf{x}}\left(\mathcal{X}^{k+1} \times \ldots \times \mathcal{X}^{k+l} \,|\, x^k, (\mathbf{u}_i)_{i\in\mathcal{I}}\right) = \int_{\mathcal{X}^{k+l-1}} \cdots \int_{\mathcal{X}^{k+1}} & P_x\left(\mathcal{X}^{k+l} \mid x^{k+l-1}, (u_i^{k+l-1})_{i\in\mathcal{I}}\right) \\ & dP_x\left(x^{k+l-1} \mid x^{k+l-2}, (u_i^{k+l-2})_{i\in\mathcal{I}}\right) \cdots \\ & dP_x\left(x^{k+1} \mid x^k, (u_i^k)_{i\in\mathcal{I}}\right) \ . \end{aligned} \tag{2.11}$$

This can be a useful probability to quantify: given a concrete sequence of control inputs (or, a sequence of control policies), how likely is the system state to remain within a certain "tube" of trajectories? Or how likely is it to satisfy a safety-critical constraint at all times?

Stochastic control policies

Rather than deterministically defining the control action chosen by an agent at each state, it may be convenient to specify a probability distribution over the set of possible control inputs. We can define a stochastic control policy as a mapping $\pi_i : \mathbb{R}^n \to \Delta\mathcal{U}_i$ that assigns to each state x^k a probability distribution over $\mathcal{U}_i$. Typically we will express such a random policy as a conditional probability measure $\pi_i(u_i^k \mid x^k)$. In the general case, we can also encode a time-varying policy as a sequence of control policies π_i^k.

While all Markov decision processes admit at least one optimal deterministic policy, stochastic control policies can be particularly useful in contexts such as reinforcement learning, in which some desirable amount of persistent exploration can be achieved through the injection of randomness. We will see an example of this in Chapter 9. In addition, stochastic

policies are a powerful modeling tool to reason about agents whose actions cannot be predicted with certainty, such as humans or unknown objects exhibiting goal-driven behavior. We will cover this case in more detail in Section 2.3, and will be making much use these policies in Chapters 7 and 8.

Risk-sensitive analysis

Operating with full probability distributions can be computationally intensive, and indeed becomes quickly intractable for high-dimensional systems. On the other hand, operating merely with the mean of the distribution would fall short of providing an adequate safety analysis, potentially overlooking highly pernicious realizations of the uncertainty carrying substantial probability. On the other hand, worst-case analysis is not always appropriate, particularly for systems that are not fully safety-critical and where it is acceptable to trade off some risk for better efficiency. This motivates a compromise between the two forms of reasoning. Risk-sensitive analysis forgoes operating with the full distribution in favor of a summary statistic, the risk measure, which is then propagated and used in decision-making. This risk measure encodes some well-defined function of the probability of undesirable outcomes and their severity. Although beyond the scope of this dissertation, some promising recent results [28] show the viability of risk-sensitive safety analysis.

2.1.5 A Note on Uncertain Perception

Finally, we cannot end this section without acknowledging an important additional source of uncertainty: perception. Throughout this thesis, unless otherwise specified, we will assume that our controller and all other agents have access to timely and accurate measurements of the state $x \in \mathbb{R}^n$ of the system. This is not a trivial assumption: perception is a fundamental component of robotics, and in particular it is an essential requirement for safety.

Fortunately, recent advances in sensing hardware (e.g. lidar) and software (e.g. deep learning techniques for computer vision), in no small part spurred by the growing interest in autonomous driving, are enabling increasingly reliable state estimation even for robotic systems operating in highly complex and unstructured environments; in other safety-critical contexts, such as the national and international airspace, next-generation technology standards, such as ADS-B, are being introduced to substantially improve detection and communication among vehicles.

The work presented in this dissertation therefore benefits directly from the growing availability of accurate perception technology for autonomous systems and builds on top of this foundation with the luxury of a certain separation of concerns. Nonetheless, when state uncertainty is significant, it directly affects safety-critical decisions and there is no question that it must be properly accounted for. Even as robotic perception continues to improve, rigorous safety analysis under noisy or imperfect measurements is likely to become an extremely important direction of research in the coming years [29].

2.2 Optimal Control and Dynamic Games

Optimal control studies the problem of optimal decision-making regarding the evolution of a dynamical system. When there are multiple decision makers with different objectives or access to information, the problem becomes a dynamic game, with the notion of optimality becoming more subtle and requiring the notion of an equilibrium.

In this overview, we will primarily focus on continuous-time optimal control and differential games, although analogous formulations exist in discrete time, and we will in fact be using them in Chapters 7–9. Important discrete-time solution approaches include model-predictive control (MPC) methods, which repeatedly plan open-loop trajectories over a finite, receding time horizon, implicitly closing the loop by frequently replanning from the most recent measured state (see for example [30] for a survey), and reinforcement learning methods, which implement different trial-and-error mechanisms to gradually improve the performance of a control policy (we discuss these methods in Section 2.4).

2.2.1 Optimal control and dynamic programming

Consider a deterministic continuous-time system of the form (2.1), evolving over a compact time interval $[0, T]$, and an objective of the form

$$\mathcal{V} : \mathbb{R}^n \times \bigcup_{t\in[0,T]} \left(\{t\} \times \mathbb{U}_t^T\right) \to \mathbb{R} \tag{2.12}$$

that assigns to every initial condition (x, t) and control signal $\mathbf{u} \in \mathbb{U}_t^T$ a scalar value encoding some quantitative property of the resulting system trajectory that we seek to maximize (or, alternatively, minimize[6]). The optimal control problem can then take the form:

$$V(x, t) := \sup_{\mathbf{u}\in\mathbb{U}_t^T} \mathcal{V}(x, t, \mathbf{u}) \ , \tag{2.13}$$

where $V : \mathbb{R}^n \times [0, T] \to \mathbb{R}$ is termed the *value* (sometimes optimal *cost-to-go* if minimizing) from initial conditions (x, t). In certain time-invariant settings the limit of this value function as $T \to \infty$ is well-defined and bounded in at least some set $\mathcal{X} \subseteq \mathbb{R}^n$, in which case we can talk about the infinite-horizon value $V(x)$.

While mathematically well defined, the optimization problem (2.13) is extremely impractical to solve directly, since this would require optimizing over the infinite-dimensional space of control signals.[7] Instead, we can exploit the central role of time in the structure of these

[6] Whether we would like to maximize or minimize this quantity is arbitrary and in fact varies depending on the field and context. For example, in reinforcement learning it is common to seek to maximize a reward, whereas in control theory the convention is to minimize a cost *except* in safety problems, where the convention is to maximize a safety margin. We will therefore be flexible as to how our objective should be treated.

[7] Even in discrete time, the problem translates into a generally non-convex, high-dimensional optimization. Finding globally or even locally optimal control signals (sequences) is a challenging computational problem, and the subject of active research in the field of nonlinear programming.

decision problems, and obtain the optimal solution without explicitly conducting the stated optimization.

Two central methods exist for tractably reasoning about the optimality of control inputs and the resulting state trajectories in nonlinear dynamical systems. The first one is the *calculus of variations*, which considers how an outcome $\mathcal{V}$ changes under local deformations of a trajectory. The core optimality result is *Pontryagin's maximum principle*, which gives a necessary condition that must be satisfied by the control input at each point of an optimal trajectory [31]. The condition is unfortunately not sufficient, so its satisfaction by a trajectory does not guarantee optimality, making its use more limited.

The other central method is dynamic programming, which reasons recursively about the optimality of strategies in backward time. The core optimality result is the Hamilton-Jacobi-Bellman equation (or simply Bellman equation in discrete time), which provides a necessary and sufficient condition that must be satisfied by the value function throughout the state space. This strong result makes Hamilton-Jacobi analysis one of the most powerful mathematical tools in optimal control and differential games, and indeed in control theory at large.

In Hamilton-Jacobi analysis, the evolution of any state function over the flow of a dynamical system can be expressed through a partial differential equation (or, in some cases that will be relevant to us, a variational inequality), whose solution will correspond to the propagated function. While Hamilton-Jacobi analysis can be used to reason about forward propagation of magnitudes like mass, electric charge or temperature in a moving fluid from some initial condition (this is known as the *advection* equation), in optimal control and differential games we are primarily interested in backward propagation from a terminal time. This backward-time reasoning based on one of the most important mathematical results in decision theory: the dynamic programming principle.

The Dynamic Programming Principle and the Bellman Equation

In the 1950s, Richard Bellman introduced dynamic programming as an approach to solve sequential decision problems based on what he termed the *principle of optimality* [32]:

> An optimal policy has the property that whatever the initial state and initial decisions are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decisions.

This had an important implication on the computation of the value of an optimal control problem: given a controlled dynamical system with no uncertainty (2.1), if the value at every state x is known at some future time $t + \delta \in (t, T]$, then it is possible to obtain the value $V(x, t)$ by simply considering the trajectory $\mathbf{x}^{\mathbf{u}}_{x,t}$ on the interval $[t, t + \delta]$.

For example, consider a typical optimal control problem with an objective of the form

$$\mathcal{V}(x, t, \mathbf{u}) = \int_t^T L\big(\mathbf{x}^{\mathbf{u}}_{x,t}(\tau), \mathbf{u}(\tau), \tau\big) d\tau + M\big(\mathbf{x}^{\mathbf{u}}_{x,t}(T)\big) \ , \tag{2.14}$$

for functions $L : \mathbb{R}^n \times \mathcal{U} \times [0,T]$ and $M : \mathbb{R}^n \to \mathbb{R}$. The value function $V : \mathbb{R}^n \times [0,T] \to \mathbb{R}$ for objective maximization is then defined as

$$V(x,t) := \sup_{\mathbf{u} \in \mathbb{U}_t^T} \int_t^T L\big(\mathbf{x}_{x,t}^{\mathbf{u}}(\tau), \mathbf{u}(\tau), \tau\big) d\tau + M\big(\mathbf{x}_{x,t}^{\mathbf{u}}(T)\big) \ . \tag{2.15}$$

The principle of optimality states that the above optimization can be arbitrarily partitioned in time, that is, for any $\delta \in [0, T-t]$:

$$V(x,t) = \sup_{\mathbf{u} \in \mathbb{U}_t^{t+\delta}} \int_t^{t+\delta} L\big(\mathbf{x}_{x,t}^{\mathbf{u}}(\tau), \mathbf{u}(\tau), \tau\big) d\tau \ + \sup_{\tilde{\mathbf{u}} \in \mathbb{U}_{t+\delta}^T} \int_{t+\delta}^T L\big(\mathbf{x}_{\tilde{x},t+\delta}^{\tilde{\mathbf{u}}}(\tau), \tilde{\mathbf{u}}(\tau), \tau\big) d\tau \ , \tag{2.16}$$

with $\tilde{x} := \mathbf{x}_{x,t}^{\mathbf{u}}(t+\delta)$. Applying (2.15) to the second term in (2.16) leads to the *integral* statement of the dynamic programming principle:

$$V(x,t) = \sup_{\mathbf{u} \in \mathbb{U}_t^{t+\delta}} \int_t^{t+\delta} L\big(\mathbf{x}_{x,t}^{\mathbf{u}}(\tau), \mathbf{u}(\tau), \tau\big) d\tau + V\big(\mathbf{x}_{x,t}^{\mathbf{u}}(t+\delta), t+\delta\big) \ . \tag{2.17}$$

Assume for now that V is everywhere differentiable. Taking the limit of (2.17) as $\delta \to 0$, we may then write the *differential* statement of the dynamic programming principle, namely the partial differential equation:

$$0 = \partial_t V + \max_{u \in \mathcal{U}} L(x,u,t) + \nabla_x V \cdot f(x,u,t) \ , \tag{2.18a}$$

with ∂_t representing the partial derivative with respect to time and ∇_x denoting the gradient with respect to the state. This has become known as the Hamilton-Jacobi-Bellman (HJB) equation, where the term $L(x,u,t) + p \cdot f(x,u,t) =: H(x,p,u,t)$ is the *Hamiltonian* and $H^*(x,p,t) := \max_{u \in \mathcal{U}} H(x,p,u,t)$ is the *optimal Hamiltonian*. If the value function is everywhere differentiable, it is the solution to the *terminal-value problem* defined by (2.18a) and the terminal condition

$$V(x,T) = M(x), \quad \forall x \in \mathbb{R}^n \ . \tag{2.18b}$$

The above optimal control problem, which has a cumulative (integral) term and a terminal term, is known as a Bolza problem. Problems with only the cumulative term L are known as Lagrange problems, and those with only the terminal term M are known as Mayer problems.[8] Table 2.2 summarizes the Hamilton-Jacobi dynamic programming equation corresponding to each of these classical types of optimal control problems.

The dynamic programming principle can also be applied to discrete-time problems. In this case, it takes the form an algebraic equation simply known as the Bellman equation (or "Bellman backup"). Table 2.3 summarizes the Bellman equation for Mayer, Lagrange, and Bolza problems.

[8] It is not hard to show, however, that any optimal control problem can be transformed into a Mayer (terminal-cost) problem by introducing an additional state variable that keeps track of the evolution of the original objective; the new objective can then simply be set to equal this state variable at the final time.

	Objective	Hamilton-Jacobi-Bellman equation
Mayer	$M\big(\mathbf{x}(T)\big)$	$-\partial_t V = \max_{u\in\mathcal{U}} \nabla_x V \cdot f(x,u,t)$ $V(x,T) = M(x)$
Lagrange	$\int_0^T L\big(\mathbf{x}(\tau),\mathbf{u}(\tau),\tau\big)d\tau$	$-\partial_t V = \max_{u\in\mathcal{U}} L(x,u,t) + \nabla_x V \cdot f(x,u,t)$ $V(x,T) = 0$
Bolza	$\int_0^T L\big(\mathbf{x}(\tau),\mathbf{u}(\tau),\tau\big)d\tau + M\big(\mathbf{x}(T)\big)$	$-\partial_t V = \max_{u\in\mathcal{U}} L(x,u,t) + \nabla_x V \cdot f(x,u,t)$ $V(x,T) = M(x)$

Table 2.2: Dynamic programming principle for classical control problems in continuous time.

	Objective	Bellman equation
Mayer	$M\big(x^K\big)$	$V^k\big(x^k\big) = \max_{u\in\mathcal{U}} V^{k+1}\big(x^{k+1}\big)$ $V^K(x^K) = M(x^K)$
Lagrange	$\sum_{k=0}^{K} L^k\big(x^k,u^k\big)$	$V^k\big(x^k\big) = \max_{u\in\mathcal{U}} L^k(x^k,u^k) + V^{k+1}\big(x^{k+1}\big)$ $V^K(x^K) = 0$
Bolza	$\sum_{k=0}^{K} L^k\big(x^k,u^k\big) + M\big(x^K\big)$	$V^k\big(x^k\big) = \max_{u\in\mathcal{U}} L^k(x^k,u^k) + V^{k+1}\big(x^{k+1}\big)$ $V^K(x^K) = M(x^K)$

Table 2.3: Dynamic programming principle for classical control problems in discrete time.

The differentiability assumption we introduced when intuitively deriving the Hamilton-Jacobi-Bellman equation constitutes a stringent limitation. Even in the seemingly benign Bolza case (2.15), there is no reason to suppose that V should be everywhere differentiable (or even continuous, unless additional assumptions are placed on $M : \mathbb{R}^n \to \mathbb{R}$). As we will see later on, safety analysis typically results in value functions whose derivatives are not well-defined everywhere.

The existence of points of non-differentiability in certain value functions should come as no surprise, since system trajectories themselves may not be smooth (recall that Carathéodory solutions, while continuous, only satisfy the dynamical ordinary differential equation at *almost all times*, which allows sharp deflections). Fortunately, there is an extended solution concept for partial differential equations that can be thought of as analogous to Carathéodory solutions for ordinary differential equations.

2.2.2 Viscosity Solutions

In the 1980s, mathematicians Crandall and Lions [33] introduced a novel solution concept for Hamilton-Jacobi partial differential equations, which they named a *viscosity solution*. Remarkably their solution concept did not only extend the classical one to allow non-differentiability, but it could be proved that it in fact provided the desired result for problems in both physics and optimal control. This immediately captured the attention of many mathematicians in the field. In particular, Evans and Souganidis [34] provided a relatively simple proof that the viscosity solution to (2.18) was in fact the value function (2.15).

The key idea behind a viscosity solution is to relax the global differentiability condition. A viscosity solution may not be differentiable everywhere[9]: at all those points where it is differentiable, the viscosity solution satisfies the partial differential equation in the classical sense; at points of non-differentiability, it satisfies a relaxed condition that can be seen as a subdifferential and superdifferential property [35].

Formally, we introduce the following definition.

Definition 2.1 (Viscosity solution)**.** *Consider a continuous Hamiltonian* $H : \mathbb{R}^n \times \mathbb{R}^n \times [0,T] \to \mathbb{R}$ *and a bounded, uniformly continuous terminal value* $M : \mathbb{R}^n \to \mathbb{R}$*. A bounded, uniformly continuous function* $V : \mathbb{R}^n \times [0,T] \to \mathbb{R}$ *is a viscosity solution of the Hamilton-Jacobi partial differential equation*

$$\partial_t V + H(x, \nabla_x V, t) = 0 \qquad \forall (x,t) \in \mathbb{R}^n \times (0,T) \tag{2.19a}$$

$$V(x,T) = M(x) \qquad \forall x \in \mathbb{R}^n \tag{2.19b}$$

if it satisfies (2.19b) *and the following two conditions:*

(a) ***Subsolution.*** *For any continuously differentiable function* $\psi \in C^1(\mathbb{R}^n \times (0,T))$*, if* $V - \psi$ *attains a local maximum at* $(x_0, t_0) \in \mathbb{R}^n \times (0,T)$*, then*

$$\partial_t \psi + H(x_0, \nabla_x \psi, t_0) \geq 0 \ . \tag{2.20}$$

(b) ***Supersolution.*** *For any continuously differentiable function* $\psi \in C^1(\mathbb{R}^n \times (0,T))$*, if* $V - \psi$ *attains a local minimum at* $(x_0, t_0) \in \mathbb{R}^n \times (0,T)$*, then*

$$\partial_t \psi + H(x_0, \nabla_x \psi, t_0) \leq 0 \ . \tag{2.21}$$

Intuitively, the *subsolution* and *supersolution* conditions state that, anywhere we can locally "touch" V—from above and from below respectively—with continuously differentiable functions ψ, these functions must locally satisfy the Hamilton-Jacobi equation relaxed as a one-sided inequality.

[9] In practice, our solutions will usually be differentiable *almost everywhere*, i.e. except for on a set of zero Lebesgue measure. This will typically correspond to hypersurfaces on which the value function presents a "ridge".

Note that anywhere where V itself is continuously differentiable, we may always choose ψ in (2.20) and (2.21) so that locally $\psi = V$, and therefore it must be that V locally satisfies the Hamilton-Jacobi inequality in both directions, that is, it must satisfy the Hamilton-Jacobi equation in the classical sense. This means that the viscosity solution is only "exempt" from satisfying the Hamilton-Jacobi equation at points where it is not continuously differentiable.

Crandall, Evans, and Lions also showed in 1984 [35] that viscosity solutions can be equivalently characterized through properties of the *superdifferential* and *subdifferential* of V, as defined below. Readers familiar with these concepts, may find that this alternative characterization provides useful additional intuition on viscosity solutions.

Definition 2.2 (Superdifferential and subdifferential)**.** *Let $V : \mathbb{R}^m \to \mathbb{R}$ and $z_0 \in \mathbb{R}^m$. The superdifferential of V at $z_0 \in \mathbb{R}^m$, denoted $D^+V(z_0)$, is the set*

$$D^+V(z_0) := \left\{p_0 \in \mathbb{R}^m : \limsup_{z \to z_0} \big(V(z) - V(z_0) - p_0 \cdot (z - z_0)\big)|z - z_0|^{-1} \leq 0\right\} \ .$$

Similarly, the subdifferential of V at $z_0 \in \mathbb{R}^m$, denoted $D^-V(z_0)$, is the set

$$D^-V(z_0) := \left\{p_0 \in \mathbb{R}^m : \limsup_{z \to z_0} \big(V(z) - V(z_0) - p_0 \cdot (z - z_0)\big)|z - z_0|^{-1} \geq 0\right\} \ .$$

That is, any element p_0 of the superdifferential $D^+(z_0)$ defines an affine function $\psi_{p_0}(z) := V(z_0) + p_0 \cdot (z - z_0)$ that locally stays above V around z_0; conversely, the corresponding affine function $\psi_{p_0}(z)$ for any element p_0 of the subdifferential $D^-V(z_0)$ will locally stay below V around z_0.

We then have the following characterization (see [35], Theorem 1.1 for its proof).

Proposition 2.1 (Alternative characterization of viscosity solutions)**.** *A bounded, uniformly continuous function $V : \mathbb{R}^n \times [0, T] \to \mathbb{R}$ is a viscosity subsolution of the Hamilton-Jacobi partial differential equation* (2.19a) *if, and only if,*

$$s + H(x, p, t) \geq 0 \ , \quad \forall (p, s) \in D^+(x, t) \ . \tag{2.22}$$

Similarly, V is a viscosity supersolution of (2.19a) *if, and only if,*

$$s + H(x, p, t) \leq 0 \ , \quad \forall (p, s) \in D^-(x, t) \ . \tag{2.23}$$

In this characterization, the superdifferential set can be seen as inducing a family of continuously differentiable (affine) functions that locally "touch" V form above, determining whether it meets the subsolution condition. The subdifferential set similarly defines a family of affine functions "touching" V from below, thus determining whether V is a supersolution.

Example 2.2. *Recall the controlled system* $\dot{x} = u$, $x \in \mathbb{R}$, $u \in [-1, 1]$ *from Example 2.1, and consider the problem of minimizing the terminal cost* $M(x) := |x|$ *after a time interval* $[0, T]$. *Clearly an optimal policy is to drive the system towards the origin as quickly as possible and, if reached, keep it there. It is straightforward to check that the corresponding value function* $V : \mathbb{R}^n \times [0, T] \to \mathbb{R}$ *is*

$$V(x, t) = \max\{0, |x| - (T - t)\} \ .$$

This function is not differentiable along the lines $x = \pm(T - t)$. *However, it satisfies the appropriate Hamilton-Jacobi-Bellman partial differential equation*

$$\begin{aligned} &\partial_t V + \min_{u \in [-1,1]} \nabla_x V \cdot u = 0 \\ &V(x, T) = |x| \end{aligned}$$

in the viscosity sense (in fact, it is the only continuous function that does [34]). We can check this by trying any element (p, s) *in the subdifferential of* V *at a point of no differentiability and verifying that the choice of control* $u = -\operatorname{sgn}(x)$ *satisfies* $s + p \cdot u \leq 0$.

The viscosity solution concept for partial differential equations (which, as we will see in Section 2.3 and Chapter 4, can be naturally extended to variational inequalities) can be seen as a generalization analogous to Carathéodory's solution concept for ordinary differential equations, in that it relaxes the global differentiability requirement. In fact, very much like Carathéodory solutions, viscosity solutions allow us to reason about problems involving dynamical systems in which control inputs may vary in non-smooth manners leading to trajectories that may not be differentiable everywhere but are nonetheless uniquely defined. This will be particularly important for safety analysis, where it is often the case that optimal control signals often switch instantaneously between extreme values, something that is commonly known as *bang-bang* control. The zero-measure sets where optimal trajectories may change direction abruptly are known as *singular surfaces*, and their study is central to many problems in optimal control and, especially, differential games.

While optimal control provides us with a solid theoretical foundation on which to build our analysis, it does not by itself incorporate a way to reason about systems with multiple inputs, whether representing multiple agents or system uncertainty in the form of a disturbance. For this we require *dynamic game theory* and, in the particular case of continuous-time analysis, *differential game theory*. As we will see, the existence of multiple agents introduces important subtleties into the analysis related to the *information structure*, namely the variables that agents are able to observe over time as they make decisions.

2.2.3 Robust Optimal Control and Zero-Sum Differential Games

Robust optimal control broadly considers the problem of maximizing or minimizing the worst-case value of a certain functional $\mathcal{V}$ given by the evolution of a dynamical system under some bounded model uncertainty. Therefore, we find ourselves in the differential inclusion setting (2.6), with a single controller using its input u to control the system in the face of an unknown but bounded disturbance d (in this exposition we will assume that the bound $d \in \mathcal{D}$ is uniform, and will extend the analysis to state-dependent bounds in Chapter 6). Since the control strategy must guarantee that a certain outcome (or better) will be achieved for any disturbance $d \in \mathcal{D}$, it is appropriate to reason about the *worst-case* outcome resulting from an adversarial realization of the uncertainty. This means that we are faced with a zero-sum differential game, with the controller and the disturbance as opposing *players*.

A pioneer in the field of differential games was Rufus Isaacs, who worked at the RAND corporation in the 1950s, during the same critical years as Richard Bellman. While Bellman is generally credited with the invention of dynamic programming for optimal control with a single decision-maker, Isaacs was independently working on its two-sided counterpart, in which two opposing decision-makers compete over the evolution of a controllable dynamical process [36]. Isaacs initial interest was centered around what he referred to as *games of pursuit* [37], in which one player is trying to move in such a way as to capture the other in a given environment, or more generally to drive the state of the system into a certain region, while the other tries to prevent this. His seminal work on pursuit-evasion games lay down the foundations of modern control-theoretic safety analysis.

In his work, Isaacs referred to games with a binary outcome (e.g. whether or not the pursuer captures the evader, or whether or not a safety violation occurs) as *games of kind*. The outcome here can only be one of two categories, and victory is a clearly defined concept: one player wins, the other loses. Conversely, other games may have the players attempting to make a scalar outcome as high or as low as possible (e.g. what is the closest distance reached between the pursuer and the evader, or how close does the system come to violating the constraints). These *games of degree* can have a continuum of possible outcomes and as a result do not have a clear definition of winning and losing, since in general both players could have conceivably done better or worse than they did. Isaacs noted that in general, a game of kind can be implicitly encoded through a game of degree: for example, the game of whether or not the pursuer captures the evader can be transformed into one where the pursuer is trying to make their closest distance over time as small as possible, while the evader is trying to keep it as large as it can; if the outcome of the game is below a specified capture radius then the pursuer wins the game of kind, otherwise the evader wins. Importantly, the optimal player strategies resulting from the game of degree are also optimal for the game of kind: that is, there is no better way to ensure capture than to minimize the closest distance over time, and there is no better way to avoid capture than to try to maximize it.

This conversion between the game of kind and the game of degree will be particularly crucial for us, since it is precisely what will enable us to reason about safety analysis and

reach-avoid problems, as we will see in Section 2.3.

Information, Strategies, and Value of the Game

We will consider a compact time interval $[0, T]$ for the game. Under this finite-horizon setting, optimal player strategies will typically be time-dependent, as will the outcome of the game from a given starting state. In Section 2.3, we will extend the horizon letting $T \to \infty$ to obtain the time-independent solution to our original safety problem. Therefore, let $\mathbb{U}_t^T$ and $\mathbb{D}_t^T$ contain all corresponding measurable input signals on the interval $[t, T]$ for any $t \in [0, T]$. The outcome of the game can be formalized as a functional

$$\mathcal{V} : \mathbb{R}^n \times \bigcup_{t\in[0,T]} \left(\{t\} \times \mathbb{U}_t^T \times \mathbb{D}_t^T\right) \to \mathbb{R} \tag{2.24}$$

assigning to each initial condition $(x, t) \in \mathbb{R}^n \times [0, T]$ and input signals $\mathbf{u} \in \mathbb{U}_t^T, \mathbf{d} \in \mathbb{D}_t^T$ a scalar value representing the quantity of interest that our controller is trying to maximize (or, depending on the problem, minimize).

As we saw in Section 2.1, reasoning directly in terms of feedback strategies can be problematic regarding the theoretical existence of trajectories. To avoid these issues, our analysis of solutions will be grounded in the notion of measurable time signals $\mathbf{u} \in \mathbb{U}_t^T$, $\mathbf{d} \in \mathbb{D}_t^T$. Our first attempt at defining the differential game may be to have the controller and the disturbance pick their input signals trying to make the outcome as positive or as negative as possible. We may then write:

$$\sup_{\mathbf{u}\in\mathbb{U}_0^T} \inf_{\mathbf{d}\in\mathbb{D}_0^T} \mathcal{V}(x, 0, \mathbf{u}, \mathbf{d}) \ . \tag{2.25}$$

Upon closer inspection, this formulation is not the most appropriate for most robust optimal control problems of interest. As discussed in Section 2.1, open-loop control is rarely used in automation systems, given the central importance (and common availability) of feedback. The optimization encoded by (2.25) implies that the controller must first commit to an input signal $\mathbf{u} : [0, T] \to \mathcal{U}$, and then the disturbance chooses its own $\mathbf{d} : [0, T] \to \mathcal{D}$, fully accounting for the input $\mathbf{u}$ chosen by the controller. This *open loop information structure* puts the controller in an overly difficult situation, since it does not have the ability to adapt its control input once the system begins to evolve. The disturbance has all of the necessary information when making its decision, whereas the controller has none. This conservative information pattern is only appropriate in scenarios where a fixed control input signal needs to be specified in advance of system deployment, or where state feedback information will be unavailable or intermittent at execution time.

In most practical cases, we would like to allow the controller to use something akin to a feedback policy, that is, it should be able to choose its input for any given time based on knowledge of how the system state will have evolved by then. Following [34, 38–40], we define the following strategy information pattern, which is central to robust optimal control and safety analysis.

Definition 2.3 (Non-anticipative strategies.)**.** *The set of non-anticipative strategies* $\mathfrak{D}_t^T$ *for the disturbance is the collection of maps* $\boldsymbol{\delta} : \mathbb{U}_t^T \to \mathbb{D}_t^T$ *such that the dependence of* $\boldsymbol{\delta}[\mathbf{u}]$ *on* $\mathbf{u}$ *is causal, that is:*

$$\mathfrak{D}_t^T = \{\boldsymbol{\delta} : \mathbb{U}_t^T \to \mathbb{D}_t^T \mid \forall s \in [t,T],\ \forall \mathbf{u}(\cdot), \hat{\mathbf{u}}(\cdot) \in \mathbb{U}_t^T, \\ \big(\mathbf{u}(\tau) = \hat{\mathbf{u}}(\tau) \text{ a.e. } \tau \in [t,s]\big) \Rightarrow \big(\boldsymbol{\delta}[\mathbf{u}](\tau) = \boldsymbol{\delta}[\hat{\mathbf{u}}](\tau) \text{ a.e. } \tau \in [t,s]\big)\} \ .$$

The above can be intuitively read as: $\boldsymbol{\delta}$ cannot preemptively start adapting to a change in $\mathbf{u}$ until such a change begins. With this information pattern, we can now formulate our differential game as:

$$V^-(x,t) := \inf_{\boldsymbol{\delta} \in \mathfrak{D}_t^T} \sup_{\mathbf{u} \in \mathbb{U}_t^T} \mathcal{V}\big(x,t,\mathbf{u},\boldsymbol{\delta}[\mathbf{u}]\big) \ , \tag{2.26}$$

where V^- is known as the *lower value* of the game. At first glance it may seem like we are now inverting the order of play and letting the controller be the one adapting to the disturbance. Note, however, that the disturbance is no longer choosing a control signal but a *mapping* from controller signals to disturbance signals; therefore, while the disturbance is choosing $\boldsymbol{\delta}$ before the controller chooses $\mathbf{u}$, ultimately $\mathbf{d} = \boldsymbol{\delta}[\mathbf{u}]$ will still depend on $\mathbf{u}$. In fact, had we not restricted $\mathfrak{D}$ to only include non-anticipative mappings, the formulations in (2.25) and (2.26) would be exactly equivalent, with $\mathbf{d}$ arbitrarily adapting to $\mathbf{u}$.

Restricting the disturbance's information about the choices of the controller in this way has an important interpretation that highlights its connection to feedback control. The information structure in (2.26) implies that any prefix of the controller's input signal $\mathbf{u}_1 : [0,t_1) \to \mathcal{U}$ determines (almost everywhere) the disturbance's response for that interval $\mathbf{d}_1 : [0,t_1) \to \mathcal{D}$ (which cannot depend anticipatively on future values taken by $\mathbf{u}$). Since there are only two inputs to the system, and given the existence and uniqueness results in Section 2.1, these two signals uniquely determine the trajectory $\mathbf{x}_{x,0}^{\mathbf{u}_1,\mathbf{d}_1}$ on any interval $[0,t]$ for $t \in [0,t_1)$; further, continuity of state trajectories grants $\mathbf{x}_{x,0}^{\mathbf{u}_1,\mathbf{d}_1}(t_1) = \lim_{t \to t_1^-} \mathbf{x}_{x,0}^{\mathbf{u}_1,\mathbf{d}_1}(t)$. Thus, because $\mathbf{u}$ can be chosen as a function of $\boldsymbol{\delta}$, the controller is effectively allowed to choose its input $\mathbf{u}(t_1)$ at each $t_1 \in [0,T]$ informed by the current state $\mathbf{x}_{x,0}^{\mathbf{u}_1,\mathbf{d}_1}(t_1)$, as well as the history of previous states and inputs.

This information structure for the controller is denoted *closed-loop perfect-state feedback*. As we will see in Section 2.3 and Chapter 3, in the class of problems we are considering, past history is in fact irrelevant to the decision-making of players provided the current state is known, and thus optimal solutions using *memoryless perfect-state feedback* always exist. While we are ultimately concerned with the design of feedback controllers that robustly ensure properties like safety and liveness, the mathematical machinery of differential games built on non-anticipative strategies allows us to ensure that Carathéodory trajectories and the resulting value functions are well-defined.

By allowing the disturbance to use non-anticipative strategies, we are still giving it the *instantaneous informational advantage* in the game, since at each instant it can adapt its control input to the one declared by the controller. This is consistent with the notion that even the instantaneous effect of our control input on the system may be uncertain (say, if we

have multiplicative uncertainty affecting the *gain* of our actuators), and we need to protect against the worst-case realization.

In principle, we could have given this advantage to our controller instead; this would not be realistic in most control applications, since the value of the disturbance can usually only be determined after the fact. Defining the analogous set $\mathfrak{U}_t^T$ of non-anticipative strategies $\boldsymbol{\upsilon}_t : \mathbb{D}_t^T \to \mathbb{U}_t^T$ for the controller would result in what is known as the *upper value* of the game:

$$V^+(x,t) := \sup_{\boldsymbol{\upsilon} \in \mathfrak{U}_t^T} \inf_{\mathbf{d} \in \mathbb{D}_t^T} \mathcal{V}\big(x, t, \boldsymbol{\upsilon}[\mathbf{d}], \mathbf{d}\big) \ , \tag{2.27}$$

From (2.26) and (2.27) it follows that $V^-(x,t) \leq V^+(x,t)$ everywhere. In those cases in which equality holds globally, the game is said to "have value" and $V(x,t) := V^+(x,t) = V^-(x,t)$ is simply referred to as the *value of the game*. This will in fact be the case in a large class of dynamical systems, in which the effects of the controller and disturbance inputs are decoupled in the dynamics. The result is that no player gains anything by instantaneously observing the other's current input, and as a result, the solution of the game is well-defined by simply giving both players state feedback information. This is not particularly relevant in robust optimal control settings, where it is not problematic to assume that in the worst case the system will *happen* to evolve in the least beneficial way at each instant; however, it becomes important in zero-sum differential games where both players are decision-making agents implemented (or strictly speaking approximated) by physical devices.[10]

The Tenet of Transition and the Isaacs Equation

With this information structure in place, we can introduce the two-sided version of the dynamic programming principle and the Hamilton-Jacobi-Bellman equation. In the 1950s, concurrently with Bellman's work, Rufus Isaacs identified the fundamental principle linking optimal player decisions over time, which he named the *tenet of transition* [37]:

> If the play proceeds from one position to a second and V is thought of as known at the second, then it is determined at the first by demanding that the players optimize (i.e. make minimax) the increment of V during the transition.

In the above statement, initially formulated for games of pursuit, Isaac's use of "position" should more generally be interpreted as "state". In general Bolza problems, the "increment of V" should also be weighed with the accrued Lagrangian term during the transition. This two-sided principle (which we often subsume today under the umbrella term "dynamic programming") enables us to extend the backward-time analysis of optimal control problems to zero-sum dynamic games (and thus robust optimal control problems). The analogue of the

[10] In games "without value" the upper and lower values can still be used to bound the outcome of the game where no player receives instantaneous information about each other. Concretely, each player can secure an outcome no worse than the value of the game in which the adversary uses non-anticipative strategies, by playing the *protected strategy* corresponding to that game.

one-sided dynamic programming principle (2.17) for a typical differential game with Bolza payoff

$$\mathcal{V}(x,t,\mathbf{u},\mathbf{d}) = \int_t^T L\big(\mathbf{x}_{x,t}^{\mathbf{u},\mathbf{d}}(\tau),\mathbf{u}(\tau),\mathbf{d}(\tau)\tau\big)d\tau + M\big(\mathbf{x}_{x,t}^{\mathbf{u},\mathbf{d}}(T)\big) \ , \tag{2.28}$$

is then given by

$$V^-(x,t) = \inf_{\boldsymbol{\delta}\in\mathfrak{D}_t^{t+\delta}} \sup_{\mathbf{u}\in\mathbb{U}_t^{t+\delta}} \int_t^{t+\delta} L\big(\mathbf{x}_{x,t}^{\mathbf{u}}(\tau),\mathbf{u}(\tau),\tau\big)d\tau + V^-\big(\mathbf{x}_{x,t}^{\mathbf{u},\boldsymbol{\delta}[\mathbf{u}]}(t+\delta),t+\delta\big) \ , \tag{2.29a}$$

$$V^+(x,t) = \sup_{\boldsymbol{\upsilon}\in\mathfrak{U}_t^{t+\delta}} \inf_{\mathbf{d}\in\mathbb{D}_t^{t+\delta}} \int_t^{t+\delta} L\big(\mathbf{x}_{x,t}^{\mathbf{u}}(\tau),\mathbf{u}(\tau),\tau\big)d\tau + V^+\big(\mathbf{x}_{x,t}^{\boldsymbol{\upsilon}[\mathbf{d}],\mathbf{d}}(t+\delta),t+\delta\big) \ . \tag{2.29b}$$

for the lower and upper values of the game as defined in (2.26) and (2.27) respectively.

Taking the limit of (2.29) as $\delta \to 0$, we arrive at the Hamilton-Jacobi-Isaacs partial differential equation:

$$0 = \partial_t V^- + \max_{u\in\mathcal{U}} \min_{d\in\mathcal{D}} L(x,u,t) + \nabla_x V^- \cdot f(x,u,t) \ , \tag{2.30a}$$

$$0 = \partial_t V^+ + \min_{d\in\mathcal{D}} \max_{u\in\mathcal{U}} L(x,u,t) + \nabla_x V^+ \cdot f(x,u,t) \ , \tag{2.30b}$$

for the lower and upper values respectively, in both cases with terminal condition

$$V^\pm(x,T) = M(x), \quad \forall x \in \mathbb{R}^n \ . \tag{2.30c}$$

Giving either of the players the ability to use non-anticipative strategies enables that player to "play second" at each instant, determining the instantaneous optimization of the Hamiltonian $H(x,p,u,d,t) := L(x,u,d,t) + p\cdot f(x,u,d,t)$ as being either minimax or maximin. We thus differentiate between the *lower Hamiltonian* $H^-(x,p,t) := \max_{u\in\mathcal{U}} \min_{d\in\mathcal{D}} H(x,p,u,d,t)$ and the *upper Hamiltonian* $H^+(x,p,t) := \min_{d\in\mathcal{D}} \max_{u\in\mathcal{U}} H(x,p,u,d,t)$.

In the 1980s, the work by Crandall, Lions, Evans, and Souganidis [33, 34] showed that the upper and lower values, whether or not everywhere differentiable, are the unique viscosity solutions to the appropriate version of (2.30). An important consequence of the uniqueness property is that whenever the minimax and the maximin in (2.30) coincide for each (x,t), or equivalently when the upper and lower Hamiltonians are identical, the upper and lower values must also be equal, and therefore the game has value (under feedback-only strategies). This is known as *Isaacs' condition*. Importantly, Isaacs' condition automatically holds in all problems where the different control inputs are decoupled in the dynamics f and have no coupling in the Lagrangian payoff L.

The analogous relation holds true in discrete-time dynamic games by giving one of the players the informational advantage at each discrete time step. Therefore, the dynamic programming relations summarized in Tables 2.2 and 2.3 for classical optimal control problems can be readily extended, both in continuous and discrete time, by replacing the maximization by a minimax or maximin as appropriate. As we will see in Section 2.3, zero-sum differential games with a different payoff class are central to safety analysis, as well as other problem specifications such as reach-avoid conditions.

2.2.4 Nonzero Sum Dynamic Games

We finally turn our attention to problems with multiple players, each with an associated payoff

$$\mathcal{V}_i : \mathbb{R}^n \times \bigcup_{t\in[0,T]} \left(\{t\} \times \prod_{i\in\mathcal{I}} \mathbb{U}_{i,t}^T \times \mathbb{D}_t^T\right) \to \mathbb{R} \ , \tag{2.31}$$

which maps an initial condition (x, t) and input signals $(\mathbf{u}_i \in \mathbb{U}_{i,t}^T)_{i\in\mathcal{I}}, \mathbf{d} \in \mathbb{D}_t^T$ to a scalar value encoding the quantity that the player in question is interested in maximizing (minimizing) as the system evolves under dynamics (2.3).

There are a variety of possible equilibrium solutions that can be defined for this general class of games (see [41] for a comprehensive reference). Our focus in this dissertation will be on (closed-loop) *feedback Stackelberg* (leader-follower) information structures [42, 43], which assign a well-defined ordering of players (without loss of generality, we assume this ordering corresponds to the player index $i = 1, \ldots, N$), such that player i has knowledge of the inputs chosen by each player $j < i$ at the time of choosing input u_i.

The nonzero-sum games that we will be working with in Part II of this dissertation will be formulated in discrete time. We will be interested in finding optimal (equilibrium) player strategies of the form

$$\gamma_i : \mathbb{R}^n \times \prod_{j<i} \mathcal{U}_j \times [0, t] \to \mathcal{U}_i \ . \tag{2.32}$$

This leads to a nested application of the dynamic programming principle, where each player i considers the *rational response* of following players ($j > i$) to every candidate input u_i. In general this rational response may not be unique, since a follower may be indifferent between multiple inputs; crucially, however, these inputs may have arbitrarily different consequences for the leader, which means that the leader will usually need to consider the *worst* rational response from each following player or alternatively use a *quantal response* model to reason about the probabilities of different follower choices. The complexity of solving these games generally suffers from combinatorial explosion, making it challenging to compute solutions for dynamic games with even a modest number of players.

In Chapter 8, we will introduce a two-player game-theoretic analysis using a quantal response model to reason about human decisions in the strategic context of driving. Other results in Chapter 7 approach real-time reasoning with multiple agents at the cost of eschewing the full game-theoretic treatment.

2.3 Safety Analysis

Safety-critical systems are those in which certain outcomes are extremely severe and essentially deemed unacceptable by the designers, users, or society at large. An example of this are civil aircraft, where substantial effort goes into certifying safety (or in the aerospace jargon *airworthiness*) of all critical components and the aircraft as a whole. This does not mean that the system *will never fail*, an impossible standard to uphold. Rather, safety analysis

enforces that the system will not be built in a way that makes failures possible under any reasonably expectable conditions. Importantly, safety-critical specifications are treated as *hard constraints* within the context of the system's design and operation, and not as merely desirable features that can be traded off with other objectives like time, efficiency, etc.

Given a mathematical model of a system's dynamics and a quantification of its associated uncertainty, safety analysis seeks to provide formal statements about whether—or with what probability—it is possible to prevent a specified set of critically undesirable outcomes, and what is the appropriate control strategy to ensure this.

Control-theoretic safety analysis takes in a controlled, possibly uncertain dynamical system

$$\dot{x} = f(x, u, d, t) \ , \tag{2.33}$$

together with a specification of the set of *failure states* $\mathcal{F} \subset \mathbb{R}^n$, and attempts to enforce the *safety condition* for the evolution $\mathbf{x}(\cdot)$ of the state over time:

$$\mathbf{x}(t) \notin \mathcal{F} \quad \forall t \geq t_0 \ . \tag{2.34}$$

Depending on the class of uncertainty, we may also pose the problem probabilistically. Following the discrete-time formulation as in (2.5), we may seek to enforce

$$P\left(\exists k \geq k_0 : x^k \in \mathcal{F}\right) \leq p_{\max} \ , \tag{2.35}$$

for some threshold probability $p_{\max} \in [0, 1]$.

It is important to stress that *both* robust and probabilistic guarantees are contingent on assumptions about the nature and structure of the uncertainty. In the robust case, all guarantees are built on d being bounded by $\mathcal{D}$ (or $\hat{\mathcal{D}}(x)$ in the case of state-dependent uncertainty); in the probabilistic case, all guarantees rest on d being drawn from a distribution P_d. Whether these bounds or distributions are directly specified by the system designer or learned automatically from data (through further structural assumptions that enable generalization from a finite set of observations), the practical assurances provided by these theoretical guarantees may cease to apply to the real system when its behavior fails to match the expected structure.

Finally, observe that (2.34) is *not* a condition about stability. Dynamic stability is neither a necessary nor a sufficient condition for safety understood as guaranteed avoidance of failure states.

This section will cover the fundamental concepts in safety analysis primarily from the point of view of robust optimal control and Hamilton-Jacobi reachability analysis, which will be used extensively in Chapters 3–6. Most of the key concepts have well-defined probabilistic counterparts through stochastic reachability. We will not be discussing these in detail, but direct the interested reader to [44]. A probabilistic safety condition in the form of (2.35) will be used in Chapter 7 when dealing with probabilistically quantified uncertainty about human motion.

2.3.1 State Constraints and Reachability Analysis

The central specification in the safety verification problem is the partition of the state space $\mathbb{R}^n$ into the *failure set* $\mathcal{F} \subset \mathbb{R}^n$, containing all states that are deemed unacceptable for the system, and the *constraint set* $\mathcal{K} := \mathcal{F}^{\mathsf{c}}$, where the system is therefore required to remain at all times during its operation. While the constraint and failure set are always (by definition) complements of each other, it is helpful to refer to both of them explicitly, and we will hence maintain the notation for both $\mathcal{K}$ and $\mathcal{F}$.

In the safety analysis that we will conduct in this dissertation, the constraint set $\mathcal{K}$ is assumed to be topologically closed (and therefore the failure set $\mathcal{F}$ is open); we will be making no further assumptions, e.g. regarding boundedness, connectedness, convexity, etc. This means that we can define an arbitrary subset of the state space as the constraint set, with the convention that if a state x is exactly on the boundary $\partial\mathcal{K}$ then it is *not* considered to be in violation of the constraint. In our exposition here, we will be considering a static constraint set and time-invariant dynamics. In Chapter 3, we will lift this restriction and consider safety analysis with time-varying dynamics and constraints.

Let the uncertain dynamics of the system be given by a differential inclusion in the form of (2.6) with a single control input and no time-variance, that is,

$$\dot{x} \in F(x, u) := \{f(x, u, d),\ d \in \mathcal{D}\} \ . \tag{2.36}$$

The safety condition (2.34) of a dynamical system is closely related to the notion of *reachability*. Given a specification of the failure set $\mathcal{F}$, we would like to characterize the set of all starting conditions $x \in \mathbb{R}^n$ from which the system state may eventually be driven into the failure set $\mathcal{F}$ in spite of the controller's efforts to prevent this. This is called the *backward-reachable set*[11] (BRS) of $\mathcal{F}$ (written $\mathcal{R}(\mathcal{F})$), because it contains all initial conditions that can be reached "in backward time" from states in $\mathcal{F}$.

Defining a backward reachable set for a system with control and disturbance inputs requires specifying or otherwise characterizing these inputs. We will therefore start with the simpler case of an *autonomous system*, in the dynamical systems theory sense, that is, a time-invariant system with no exogenous control or disturbance inputs ($\dot{x} = f(x)$). This can also be thought of as the result of specifying a particular feedback control policy and analyzing the resulting *closed-loop dynamics*, which are now merely state-dependent: $f_\pi(x) := f(x, \pi(x))$.

Definition 2.4 (Backward-reachable set)**.** *The backward-reachable set $\mathcal{R}(\mathcal{T})$ of a set $\mathcal{T} \subseteq \mathbb{R}^n$ under the autonomous dynamical system $\dot{x} = f(x)$ is the set of initial states $x \in \mathbb{R}^n$ from which the state trajectory will reach $\mathcal{T}$ at some future time.*

$$\mathcal{R}(\mathcal{T}) := \{x \in \mathbb{R}^n : \exists t \geq t_0, \mathbf{x}_{x,t_0}(t) \in \mathcal{T}\} \ . \tag{2.37}$$

[11] Some authors prefer the term *backward-reachable tube* to distinguish this set from the set of initial conditions for which the state will be driven to $\mathcal{F}$ at an *exact* time, rather than any future time. In this dissertation, we will use the *backward-reachable set* terminology, making any necessary clarifications explicit.

In this general context, $\mathcal{T} \in \mathbb{R}^n$ is often referred to as the *target* set. When there is a single input to the dynamics and we seek to find all the states from which the system could be driven into $\mathcal{T}$, or conversely only the states from which the system cannot be steered clear of $\mathcal{T}$, it makes sense to speak of *maximal* and *minimal* backward-reachable sets respectively. However, if there are two control inputs with opposite intent, in this case our best-effort controller accounting for the worst-case disturbance, the notion of maximal and minimal becomes ambiguous. Instead, we will try to use an intuitive and easy to remember naming convention by adopting the point of view of the controller guarding against all possible realizations of the disturbance.

Definition 2.5 (Enforceable backward-reachable set)**.** *The enforceable backward-reachable set $\overline{\mathcal{R}}(\mathcal{T})$ of a set $\mathcal{T} \subseteq \mathbb{R}^n$ under the uncertain dynamics* (2.36) *is the set of initial states $x \in \mathbb{R}^n$ from which for every non-anticipative disturbance strategy $\boldsymbol{\delta} : \mathbb{U} \to \mathbb{D}$ there exists a measurable control signal $\mathbf{u} \in \mathbb{U}$ that can steer the system trajectory into $\mathcal{T}$ at some future time.*

$$\overline{\mathcal{R}}(\mathcal{T}) := \left\{ x \in \mathbb{R}^n : \forall \boldsymbol{\delta} \in \mathfrak{D}, \exists \mathbf{u} \in \mathbb{U}, \exists t \geq t_0, \mathbf{x}_{x,t_0}^{\mathbf{u},\boldsymbol{\delta}[\mathbf{u}]}(t) \in \mathcal{T} \right\} \quad . \tag{2.38}$$

The enforceable backward-reachable set is typically useful when studying *liveness*, that is, the controller's ability to complete a desired task; it is not particularly useful in quantifying the safety condition (2.34), where we are trying to *avoid* reaching a certain target set, namely the failure set. For this, the following, converse definition is directly applicable.

Definition 2.6 (Inevitable backward-reachable set)**.** *The inevitable backward-reachable set $\underline{\mathcal{R}}(\mathcal{T})$ of a set $\mathcal{T} \subseteq \mathbb{R}^n$ under the uncertain dynamics* (2.36) *is the set of initial states $x \in \mathbb{R}^n$ from which there exists a non-anticipative disturbance strategy $\boldsymbol{\delta} : \mathbb{U} \to \mathbb{D}$ such that no measurable control signal $\mathbf{u} \in \mathbb{U}$ can prevent the system trajectory from reaching $\mathcal{T}$ at some future time.*

$$\underline{\mathcal{R}}(\mathcal{T}) := \left\{ x \in \mathbb{R}^n : \exists \boldsymbol{\delta} \in \mathfrak{D}, \forall \mathbf{u} \in \mathbb{U}, \exists t \geq t_0, \mathbf{x}_{x,t_0}^{\mathbf{u},\boldsymbol{\delta}[\mathbf{u}]}(t) \in \mathcal{T} \right\} \quad . \tag{2.39}$$

The order of quantifiers in the above definitions is important, and may seem unintuitive at first, since it may appear as though by allowing $\mathbf{u}$ to be chosen after $\boldsymbol{\delta}$ we are giving the controller an unrealistic advantage over the disturbance; this is not the case. Analogously to the treatment with differential games in Section 2.2, the disturbance is not committing to an input signal but to a *mapping* that will still allow it to adapt to the controller's input signal; the restriction that we impose is that this mapping must be non-anticipative (causal). Given this, we must force the disturbance to choose its non-anticipative strategy *before* the control signal has been declared; failure to do so would allow the disturbance to "cheat" and choose the mapping $\mathbf{u} \mapsto \mathbf{d}$ already accounting for what $\mathbf{u}$ will be at all times, which would amount to letting it choose its signal $\mathbf{d}$ with full prior knowledge of the entire $\mathbf{u}$. Therefore the order of quantifiers prevents the disturbance from effectively adapting ahead of time to inputs that the controller has not yet provided.

When applied to the physical systems that we are interested in analyzing, these definitions effectively mean that from any state $x \in \overline{\mathcal{R}}(\mathcal{T})$ (respectively $x \notin \underline{\mathcal{R}}(\mathcal{T})$) we can construct a *feedback control policy* that, for all possible realizations of the dynamic uncertainty, guides the state to the target $\mathcal{T}$ (or, respectively, keeps it away from $\mathcal{T}$). While the rigorous definitions and mathematical treatment (cf. [41, 45]) are formulated in terms of measurable signals and non-anticipative strategies in order to avoid the possibility of mathematical artifacts of the kind of Example 2.1, it will be useful to keep in mind the connection to robust state feedback, since ultimately our goal is the implementation of safety-preserving controllers.

The following result sheds some light on this connection by establishing a sufficient condition for the system state to be inside the enforceable backward-reachable set or outside the inevitable backward-reachable set based on the existence of the appropriate, well-behaved feedback control policy.

Proposition 2.2 (Sufficient feedback policy condition for backward reachability)**.** *Let there exist a time-invariant feedback control policy $\pi : \mathbb{R}^n \to \mathcal{U}$ such that, for any measurable disturbance signal $\mathbf{d} \in \mathbb{D}$, there is a unique (Carathéodory) state trajectory $\mathbf{x}$ and an associated measurable control signal $\mathbf{u}(\cdot) = \pi\big(\mathbf{x}(\cdot)\big)$ that satisfy*

$$\begin{aligned} \dot{\mathbf{x}}(t) &= f\big(\mathbf{x}(t), \pi\big(\mathbf{x}(t)\big), \mathbf{d}(t), t\big), \ \ a.e.\ t \geq 0 \ , \\ \mathbf{x}(t_0) &= x \ , \end{aligned}$$

for some $x \in \mathbb{R}^n$. Then, the following hold.

(a) If π is such that for every possible realization of the uncertainty $\mathbf{d} \in \mathbb{D}$, the system trajectory starting at x eventually reaches $\mathcal{T}$, then x is in the enforceable backward-reachable set $\overline{\mathcal{R}}(\mathcal{T})$.

(b) If π is such that for every possible realization of the uncertainty $\mathbf{d} \in \mathbb{D}$, the system trajectory starting at x remains outside $\mathcal{T}$, then x is outside of the inevitable backward-reachable set $\overline{\mathcal{R}}(\mathcal{T})$.

Proof. Since Carathéodory trajectories are continuous, we have that, for any $\mathbf{d} \in \mathbb{D}$, the state reached by the resulting trajectory $\mathbf{x}^{\pi,\mathbf{d}}_{x,t_0}$ at any time $t > t_0$ is $\mathbf{x}^{\pi,\mathbf{d}}_{x,t_0}(t) = \lim_{\tau \to t^-} \mathbf{x}^{\pi,\mathbf{d}}_{x,t_0}(\tau)$. Denoting by $\mathbf{d}_{[t_0,\tau]}$ the restriction of $\mathbf{d}$ to the time interval $[t_0, \tau])$, the control signal (by hypothesis measurable) resulting from policy π under disturbance can be constructed as

$$\mathbf{u}(t) := \pi\Big(\lim_{\tau \to t^-} \mathbf{x}^{\pi,\mathbf{d}_{[t_0,\tau]}}_{x,t_0}(t) \Big) \ ,$$

which, at each time $t > 0$ depends only on values taken by the disturbance signal on $[t_0, t)$; at exactly $t = t_0$, we simply have $\mathbf{u}(t_0) := \pi(x)$.

Now, for any non-anticipative mapping $\boldsymbol{\delta} : \mathbb{U} \to \mathbb{D}$, the signal $\boldsymbol{\delta}[\mathbf{u}]$ on any interval $[t_0, \tau]$ must be determined (almost everywhere) by the values of $\mathbf{u}$ on $[t_0, \tau]$, and we can therefore

define the corresponding "truncated" mapping $\boldsymbol{\delta}_{[t_0,\tau]} : \mathbb{U}_{t_0}^{\tau} \to \mathbb{D}_{t_0}^{\tau}$. Then, given an arbitrary non-anticipative mapping $\boldsymbol{\delta}$, we can always construct a control signal $\mathbf{u}$ of the form:

$$\mathbf{u}(t) := \pi\Big(\lim_{\tau \to t^-} \mathbf{x}_{x,t_0}^{\pi, \boldsymbol{\delta}_{[t_0,\tau]}[\mathbf{u}_{[t_0,\tau]}]}(t) \Big) ,$$

again with $\mathbf{u}(t_0) := \pi(x)$. By hypothesis, the resulting trajectory $\mathbf{x}_{x,t_0}^{\mathbf{u},\boldsymbol{\delta}[\mathbf{u}]}$ is well-defined and identical to the trajectory $\mathbf{x}_{x,t_0}^{\pi,\mathbf{d}}$ for $\mathbf{d} = \boldsymbol{\delta}[\mathbf{u}]$.

If (a) holds, then this trajectory eventually reaches the target $\mathcal{T}$. This implies condition (2.38) and therefore $x \in \overline{\mathcal{R}}(\mathcal{T})$.

On the other hand, if (b) holds, then the trajectory remains outside of the target $\mathcal{T}$ for all time, which implies condition (2.39) and thus $x \notin \underline{\mathcal{R}}(\mathcal{T})$. □

This result falls short of a full characterization of robust reachability purely in terms of feedback control policies, and the game-theoretic formulation remains the appropriate analytical approach. While we may not be able to prove that such well-behaved feedback control strategies *must* exist in the general case, we can use viscosity solution theory to reason about how to construct discrete-time feedback policies that arbitrarily approach the optimal outcome of the differential game as the time step becomes small. In Chapter 3 (specifically in Lemma 3.3), we will obtain further insight into how appropriately-constructed control policies of this form can robustly enforce system behavior arbitrarily close to the game-theoretic one.

This has powerful practical implications: in reality we can never construct an *instantaneous* feedback control policy, and most controllers today are implemented on digital computers, only modifying the value of the control input at a finite rate, once every *control cycle*. This results in what is commonly referred to as *sampled-data systems*. Fortunately, the viscosity solutions to the reachability problems that we are concerned with here have excellent numerical properties, enabling accurate approximation through numerical methods as well as near-optimal implementation by sampled-data controllers. As we will see later in this section, Hamilton-Jacobi analysis allows us to not only compute these reachable sets as the solution to the corresponding differential games, but also implicitly derive feedback control policies enforcing the desired property. Throughout this dissertation, we will tend to work with continuous-time feedback policies for simplicity—however, the corresponding results can be extended (typically with slightly more conservative bounds) to digitally-implemented controllers. For the interested reader, an analysis of sampled-data implementations of safety-preserving control policies can be found in [46, 47].

By construction, then, the inevitable backward-reachable set of the *failure* set, that is, $\underline{\mathcal{R}}(\mathcal{F})$, contains all states from which the safety condition cannot be enforced by the controller, and safety violations may occur under the uncertain dynamics—therefore, we refer to $\underline{\mathcal{R}}(\mathcal{F})$ as the *unsafe set*. Note that, by Definition 2.6, $\mathcal{F} \subseteq \underline{\mathcal{R}}(\mathcal{F})$, that is, all states currently in violation of the safety constraints automatically imply a failure of the safety condition (2.34). Conversely, the complement of $\underline{\mathcal{R}}(\mathcal{F})$ contains all states from which there

exists some course of action for the controller to ensure that the system will remain in $\mathcal{K}$ for all time, and is therefore called the *safe set*.

Definition 2.7 (Safe set)**.** *The safe set Ω of a system governed by the dynamical inclusion* (2.36) *with respect to a failure set $\mathcal{F} \subseteq \mathbb{R}^n$ is the set of initial states $x \in \mathbb{R}^n$ from which, for every non-anticipative disturbance strategy $\boldsymbol{\delta} \in \mathfrak{D}$, there exists a control signal $\mathbf{u} \in \mathbb{U}$ that can keep the system trajectory outside of $\mathcal{F}$ for all future time.*

$$\Omega := \left\{ x \in \mathbb{R}^n : \forall \boldsymbol{\delta} \in \mathfrak{D}, \exists \mathbf{u} \in \mathbb{U}, \forall t \geq t_0, \mathbf{x}_{x,t_0}^{\mathbf{u},\boldsymbol{\delta}[\mathbf{u}]}(t) \notin \mathcal{F} \right\} \ . \tag{2.40}$$

It can be readily checked by inspection of Definitions 2.7 and 2.6 that $\Omega = \underline{\mathcal{R}}(\mathcal{F})^{\mathsf{c}}$. In addition, we have that $\Omega \subseteq \mathcal{K}$, that is, any states from which the safety condition (2.34) can be maintained must also necessarily satisfy the safety constraint at the current time.

The safe set Ω also has corresponding definitions in viability theory [27]. For an autonomous dynamical system (i.e. one with no control or disturbance inputs), Ω is the largest (positively) invariant set contained in $\mathcal{K}$, and is referred to as the *invariance kernel* of the constraint set, $\mathrm{Inv}(\mathcal{K})$. In a controlled system with no uncertainty, Ω is the largest (positively) controlled invariant set contained in $\mathcal{K}$, denoted the *viability kernel* $\mathrm{Viab}(\mathcal{K})$. Finally, in the general case given in (2.40), Ω is the largest (positively) robust controlled invariant set contained in $\mathcal{K}$, called the *discriminating kernel* $\mathrm{Disc}(\mathcal{K})$.

The positive invariance of the safe set may not be obvious at first glance, but it follows from its definition. We can formalize this through the following proposition, formulating robust controlled invariance in analogous game-theoretic terms as reachability and safety.

Proposition 2.3 (Invariance of the safe set)**.** *The safe set Ω is a (positively) robust controlled invariant set under the uncertain dynamics* (2.36)*, that is,*

$$\forall x \in \Omega, \forall \boldsymbol{\delta} \in \mathfrak{D}, \exists \mathbf{u} \in \mathbb{U}, \forall t \geq t_0, \mathbf{x}_{x,t_0}^{\mathbf{u},\boldsymbol{\delta}[\mathbf{u}]}(t) \in \Omega, \forall t \geq t_0 \ .$$

Proof. We can prove this statement by contradiction. Suppose the safe set Ω is not in fact a robust controlled invariant set, that is, there is some state $x \in \Omega$ from which some non-anticipative disturbance strategy $\boldsymbol{\delta} \in \mathfrak{D}$ is able to eventually drive the trajectory out of Ω for every control signal $\mathbf{u}$. Then, there exist some state $\tilde{x}[\mathbf{u}] \notin \Omega$ and $\tau[\mathbf{u}] > t_0$ for which $\mathbf{x}_{x,t_0}^{\boldsymbol{\delta}[\mathbf{u}],\mathbf{u}}(\tau) = \tilde{x}$. From this $\tilde{x}[\mathbf{u}]$, there exists a new non-anticipative strategy $\boldsymbol{\delta}'[\mathbf{u}]$ such that, for all subsequent control signals $\mathbf{u}'$, the state will eventually reach the failure set $\mathcal{F}$. This means that we can construct a new non-anticipative strategy $\boldsymbol{\delta}''$ such that, for each $\mathbf{u}$, it produces the disturbance signal $\boldsymbol{\delta}[\mathbf{u}]$ for all $t \in [t_0, \tau[\mathbf{u}])$ (which will drive the trajectory from x to $\tilde{x}[\mathbf{u}] \notin \Omega$) and subsequently switches to the non-anticipative strategy $\boldsymbol{\delta}'[\mathbf{u}]$ (thereby driving the trajectory into $\mathcal{T}$). By Definition 2.7, the existence of this non-anticipative strategy $\boldsymbol{\delta}'' \in \mathfrak{D}$ means that $x \notin \Omega$, which is a contradiction. □

This invariance property will be of central importance in the design of *least-restrictive* supervisory schemes, which can allow arbitrary system behavior while on the interior of the

safe set but will enforce the safety-preserving action at (or near) its boundary to ensure its invariance and therefore keep the system from ever violating the constraints. Such supervisory schemes can be used in combination with data-driven methods to enable safe learning-based control, which will be the focus of Chapter 6.

It is important to insist early on the distinction between states that are *safe* and states that are merely inside the constraint set. As we have seen, $\Omega \subseteq \mathcal{K}$, and in most systems of practical relevance (from power grids to aircraft) the inclusion is strict, that is, there will usually exist states that are currently violation-free but are nonetheless not safe—therefore, it is also important to distinguish between states that are *unsafe*, i.e. states from which the system may inevitably violate the constraints in the future, and *failure* states, which are intrinsically in violation of the constraints. Table 2.4 summarizes the relation between the four sets.

Failure set	$\mathcal{F} \subset \mathbb{R}^n$	All states that must be avoided.
Constraint set	$\mathcal{K} = \mathcal{F}^\mathsf{c}$	Contains safe and unsafe states.
Safe set	$\Omega \subseteq \mathcal{K}$	System can be robustly kept out of the failure set.
Unsafe set	$\underline{\mathcal{R}}(\mathcal{F}) = \Omega^\mathsf{c}$	System may reach the failure set despite control efforts.

Table 2.4: Failure, constraint, safe, and unsafe sets.

Consider the quadrotor in Figure 2.1 flying with the safety constraint of not colliding with the ground. Is it safe? Surely if it is flying above the ground it is not *currently* in violation of the constraint. However, what is its current vertical velocity? What is its angular attitude? If the vehicle is currently in a fast dive, or rotated to a 90° bank angle, it may very well be that its fate has already been sealed no matter how skillfully the pilot, human or automated, attempts its recovery.

In many engineering problems, safety constraints such as "do not collide" can be specified and encoded by system designers without difficulty (say, "altitude must be no less than 1 cm": $\mathcal{K} := \{x : x_h \geq 0.01 \text{ m}\}$). Once this has been done, determining whether a particular system state satisfies these constraints is a straightforward operation. Conversely, determining whether a given state is *safe* is much more involved, since it requires reasoning not only about whether the current state is violating the constraints, but also about whether future states might; since these future states will in turn depend on what inputs our controller decides to inject and how the uncertain dynamics respond, this is far from a trivial calculation. Checking whether a state is safe and finding a suitable control policy to enforce this safety are intimately coupled problems, as made apparent by Definition 2.7; therefore, the safety analysis we seek to carry out is not only *descriptive*, but also *prescriptive*. Consistently, it is useful to study the safety problem in the framework of optimal control. Our focus will be on what has arguably been the most general and broadly applicable methodology for formulating and computing safety analysis: robust optimal control through the level-set method. We will review alternative methodologies at the end of this section.

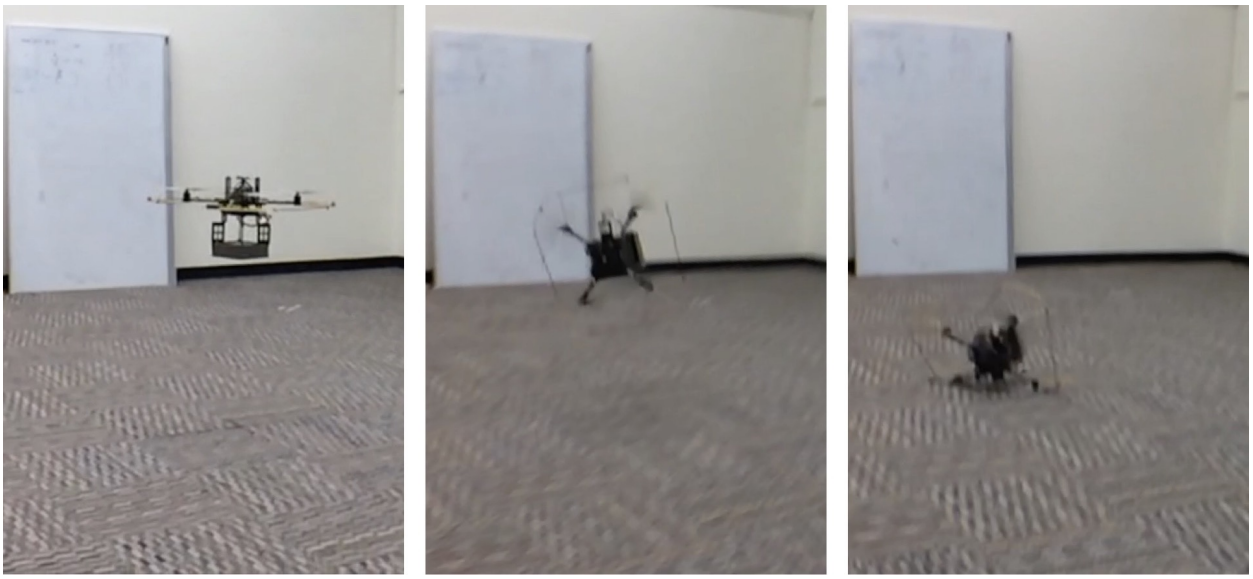


Figure 2.1: An Ascending Technologies Hummingbird quadrotor in a safe (left), unsafe (center), and failure (right) state. From the unsafe state, there is no physically realizable control input that can prevent a ground collision (failure) from taking place.

2.3.2 The Level-Set Approach

Before formulating our optimal control problem, we need to introduce an important mathematical technique that will allow us to translate relations between system trajectories and regions of the state space into functionals that can be made amenable to optimization. We can implicitly characterize $\mathcal{K}$ as the zero superlevel set of a Lipschitz *surface function* $g : \mathbb{R}^n \to \mathbb{R}$:

$$x \in \mathcal{K} \iff g(x) \geq 0 \ . \tag{2.41}$$

This function always exists, since we can simply choose $g(x)$ to be the signed distance[12] to the failure set, $s_{\mathcal{F}}(x)$, which is Lipschitz continuous by definition, and it is straightforward to check that it satisfies (2.41). We often refer to g as the *safety margin* function.

To express whether a given trajectory *ever* violates the constraints, let the functional $\mathcal{V} : \mathbb{R}^n \times \mathbb{U} \times \mathbb{D} \to \mathbb{R}$ assign to each initial state x and input signals $\mathbf{u}$, $\mathbf{d}$ the lowest value of $g(\cdot)$ achieved by trajectory $\mathbf{x}_x^{\mathbf{u},\mathbf{d}}$ over all times $t \geq 0$:

$$\mathcal{V}\big(x, \mathbf{u}, \mathbf{d}\big) := \inf_{t \geq 0} g\big(\mathbf{x}_x^{\mathbf{u},\mathbf{d}}(t)\big) \ . \tag{2.42}$$

This outcome $\mathcal{V}$ will be strictly smaller than zero if there exists any $t \in [t_0, \infty)$ at which the trajectory leaves the constraint set, and will be nonnegative if the system remains in

[12]For any nonempty set $\mathcal{M} \subset \mathbb{R}^m$, the signed distance function $s_{\mathcal{M}} : \mathbb{R}^m \to \mathbb{R}$ is defined as $\inf_{y \in \mathcal{M}} |z - y|$ for points $z \in \mathcal{M}^{\mathsf{c}}$ and $-\inf_{y \in \mathcal{M}^{\mathsf{c}}} |z - y|$ for points $z \in \mathcal{M}$, where $|\cdot|$ denotes a norm on $\mathbb{R}^m$.

the constraint set for all of $t \geq t_0$. Denoting $\mathcal{V}^{\mathbf{u},\mathbf{d}}(x) := \mathcal{V}\big(x, \mathbf{u}, \mathbf{d}\big)$, the following statement follows from (2.41) and (2.42) by construction.

Proposition 2.4. *The set of states x from which the system trajectory $\mathbf{x}_x^{\mathbf{u},\mathbf{d}}$ under given inputs $\mathbf{u} \in \mathbb{U}, \mathbf{d} \in \mathbb{D}$ will remain in the constraint set $\mathcal{K}$ at all times $t \geq 0$ is equal to the zero superlevel set of $\mathcal{V}^{\mathbf{u},\mathbf{d}}$:*

$$\{x \in \mathbb{R}^n : \forall t \geq 0,\ \mathbf{x}_x^{\mathbf{u},\mathbf{d}}(t) \in \mathcal{K}\} = \{x \in \mathbb{R}^n : \mathcal{V}^{\mathbf{u},\mathbf{d}}(x) \geq 0\}.$$

Note that, while this condition looks somewhat similar to Definition 2.7, the above set is not generally identical to Ω, since $\mathbf{u}$ and $\mathbf{d}$ here are an arbitrary pair of signals.

Proposition 2.4 allows us to analyze the safety problem as a robust optimal control problem where the controller is trying to maximize the outcome under the worst-case disturbance. By using $\mathcal{V}$ we are encoding the safety *game of kind* through an auxiliary *game of degree*. Our original formulation did not involve any preferences about how much margin was maintained between the state trajectory and the failure set, as long as this margin never became negative. That is, the safety condition (2.34) specifies a logical property, not a scalar quantity. In contrast, here the controller will attempt to make the outcome $\mathcal{V}$ as positive as possible. Nevertheless, as we will see, the solution to the auxiliary problem can easily be translated back into the solution to our original safety problem.

2.3.3 Hamilton-Jacobi Safety Analysis

To compute a *safe set* and an associated *safety policy*, we formulate a zero-sum differential game whose outcome is given by the functional $\mathcal{V}\big(x, \mathbf{u}, \mathbf{d}\big)$ introduced in (2.42), negative for those trajectories $\mathbf{x}_x^{\mathbf{u},\mathbf{d}}$ that at some point violate the constraints $\mathcal{K}$.

In the robust safety problem, the controller seeks to maximize the outcome of the game, while the disturbance tries to minimize it: that is, the disturbance is trying to drive the system out of the constraint set, and the controller wants to prevent it from succeeding.

As in the differential games of Section 2.2, and consistent with Definition 2.7, we give the disturbance the instantaneous informational advantage by allowing it to use non-anticipative strategies $\boldsymbol{\delta} : \mathbb{U} \to \mathbb{D}$. The *safety value* of the game is then given by:

$$V(x) := \inf_{\boldsymbol{\delta} \in \mathfrak{D}} \sup_{\mathbf{u} \in \mathbb{U}} \mathcal{V}\big(x, \mathbf{u}, \boldsymbol{\delta}[\mathbf{u}]\big)\ , \tag{2.43}$$

which corresponds to the *lower value* introduced in Section 2.2. The following classical result follows from Proposition 2.4.

Proposition 2.5. *The safe set Ω corresponding to constraints $\mathcal{K}$ and dynamics* (2.36) *is the zero superlevel set of the value function V:*

$$\Omega = \{x \in \mathbb{R}^n : V(x) \geq 0\}\ .$$

It has been shown that the value function for games with *minimum payoff* of the form of (2.42) can be characterized as the unique viscosity solution to a variational inequality involving an appropriate Hamiltonian [48]; an alternative formulation involves a modified partial differential equation [49]. In a finite-horizon setting, with the game taking place over the compact time interval $[0, T]$, the lower value function $V^-(x,t)$ can be computed by solving the terminal value problem with the Hamilton-Jacobi-Isaacs variational inequality:

$$0 = \min\left\{g(x) - V^-(x,t), \partial_t V^-(x,t) + \max_{u\in\mathcal{U}} \min_{d\in\mathcal{D}} \nabla_x V^-(x,t) f(x,u,d)\right\} \tag{2.44a}$$

$$V^-(x,T) = g(x) \ . \tag{2.44b}$$

The safety margin g is said to act as an *obstacle* in the Hamilton-Jacobi-Isaacs equation, because it inhibits the propagation of the value function through the term $g(x) - V^-(x,t)$ in the minimum. In Chapter 3, we will prove a general *double-obstacle* Hamilton-Jacobi-Isaacs equation which directly implies (2.44) as a special case.

As long as there exists a nonempty safe set in the problem, $V^-(x,t)$ becomes independent of t inside of this set as $T \to \infty$. We accordingly drop the dependence on t and recover $V(x) = \lim_{t\to-\infty} V^-(x,t)$ as defined in (2.43).

Definition 2.8. *The* optimal safe policy π^* *is the solution to the optimization:*[13]

$$\pi^*(x) = \arg\max_{u\in\mathcal{U}} \min_{d\in\mathcal{D}} \nabla_x V(x) f(x,u,d) \ .$$

Policy $\pi^*(x)$ attempts to drive the system to the safest possible state always assuming an adversarial disturbance. Going forward, we will generally not be concerned with potential existence issues of Carathéodory trajectories under π^*, since in practice (as we will more formally establish in Chapter 3) we can always obtain well-defined trajectories arbitrarily close to the robust optimal outcome by using discrete-time approximations of π^* with a sufficiently small time step.

2.3.4 Least-Restrictive Supervisory Control

As long as $d \in \mathcal{D}$, one can allow the system to execute any desired control while in the interior of Ω, as long as the safety preserving action $\pi^*(x)$ is taken whenever the state reaches a neighborhood of the boundary $\partial\Omega$; the system is then guaranteed to remain inside Ω for all time. This safety-preserving policy can be used in conjunction with an arbitrary control policy $\pi_g(x)$ (typically performance-driven), to produce a *least-restrictive* supervisory control scheme:

$$\pi(x) = \begin{cases} \pi_g(x) & \text{if } V(x) \geq \epsilon > 0 \ , \\ \pi^*(x) & \text{otherwise} \ , \end{cases} \tag{2.45}$$

[13] While in general the solution need not be unique, we can always choose one element of the arg max set arbitrarily. Therefore we will assume for simplicity a policy $\pi^* : \mathbb{R}^n \to \mathcal{U}$ uniquely mapping states to control inputs.

for some (typically small) $\epsilon > 0$.

Remark 2.1. *Rather than imposing the optimal safe action $\pi^*(x)$, it is in fact sufficient to project the desired $\pi_g(x)$ onto the set of control inputs that guarantee nonnegative local evolution of V for all $d \in \mathcal{D}$. However, $\pi^*(x)$ results in the greatest predicted increase in value, which is desirable under model uncertainty, as we will discuss in detail in Chapter 6.*

2.3.5 Reach-Avoid Problems

Beyond the problem of reaching a target set in the state space or avoiding some failure set of forbidden states, we may often care about defining more complex requirements by combining safety and liveness properties. The canonical example of this are *reach-avoid problems*, in which the controller is required to drive the system into a desired target set $\mathcal{T} \subset \mathbb{R}^n$ while avoiding a failure set $\mathcal{F} \subset \mathbb{R}^n$ (or equivalently remaining inside a constraint set $\mathcal{K} = \mathcal{F}^\mathsf{c}$) at all previous times.

Similarly to the safety condition (2.34), the *reach-avoid condition* for a state trajectory $\mathbf{x}$ can be formalized as

$$\exists \tau \in [t, T] \;\; | \;\; \mathbf{x}(\tau) \in \mathcal{T} \;\; \wedge \;\; \forall s \in [t, \tau], \mathbf{x}(\tau) \in \mathcal{K} \;\; . \tag{2.46}$$

For an uncertain dynamical system (2.36), we can define the reach-avoid differential game as a *game of kind* in which the controller will try to enforce (3.4) in spite of the worst-case realization of the disturbance. Adopting the usual convention of equipping the disturbance with non-anticipative strategies, we can define the controller's victory domain (or winning set) of starting states.

Definition 2.9 (Enforceable reach-avoid set)**.** *The enforceable reach-avoid set $\mathcal{RA}(\mathcal{T}, \mathcal{K})$ of a set $\mathcal{T} \subseteq \mathbb{R}^n$ under the uncertain dynamics* (2.36) *is the set of initial states $x \in \mathbb{R}^n$ from which for every non-anticipative disturbance strategy $\boldsymbol{\delta} : \mathbb{U} \to \mathbb{D}$ there exists a measurable control signal $\mathbf{u} \in \mathbb{U}$ that can steer the system trajectory into $\mathcal{T}$ at some future time without leaving $\mathcal{K}$ at any previous time.*

$$\begin{aligned} \mathcal{RA}(\mathcal{T}; \mathcal{K}) := \Big\{ & x \in \mathbb{R}^n : \forall \boldsymbol{\delta} \in \mathfrak{D}, \exists \mathbf{u} \in \mathbb{U}, \\ & \exists \tau \in [t, T] \;\; | \;\; \mathbf{x}_{x,t_0}^{\mathbf{u}, \boldsymbol{\delta}[\mathbf{u}]}(\tau) \in \mathcal{T} \;\wedge\; \forall s \in [t, \tau], \mathbf{x}_{x,t_0}^{\mathbf{u}, \boldsymbol{\delta}[\mathbf{u}]}(\tau) \in \mathcal{K} \Big\} \;\; . \end{aligned} \tag{2.47}$$

Similar to the case with reachability and safety problems, it is possible to encode this differential game through an auxiliary *game of degree*. We will address reach-avoid games in more depth in Chapter 3, where we will also extend the Hamilton-Jacobi machinery to efficiently solve such problems in time-varying settings.

2.3.6 Computational Methods

The safe set and optimal safety policy (and more generally, the backward-reachable set and reach-avoid set corresponding to an optimal control problem or differential game) can be numerically computed through a suite of partial differential equation solvers, which can approximate the value function to an arbitrary degree of precision, thanks to the desirable numerical properties presented by viscosity solutions [50].

In particular, a rich suite of numerical methods have specifically been developed for the solution of Hamilton-Jacobi equations [51–53], based on accurate approximations of spacial and time derivatives and efficient integration schemes. We directly benefit from these in safety and reachability computations carried out throughout this dissertation.

In addition to Hamilton-Jacobi analysis, a number of different mathematical approaches have been proposed to compute reachable sets and perform safety verification. Certain methods for systems with linear dynamics can be used to compute or over-approximate the reachable set of interest through parametrized geometric representations like ellipsoids [54–56] and polytopes [55, 57]. Many methods in this class require additional convexity assumptions on the constraint or target sets—in some cases, representing them through their support functions [55, 58].

For nonlinear dynamics, the computation of reachable sets must generally be done numerically, often through grid-based methods. Some recent verification approaches based on simulation [59] or logical solvers [60] can be used in nonlinear systems with no control input (or with a prescribed control policy) to locally check reachability from specified initial conditions. These local approaches do not compute reachable sets or lend themselves to controller synthesis, which limits their use as prescriptive tools. Nevertheless these techniques may become particularly powerful in conjunction with approaches based on approximate dynamic programming and reinforcement learning, which prescribe best-effort control policies without intrinsically verified properties (we will expand on this idea in Chapter 4 and then revisit it in Chapter 10).

2.4 Learning-Based Control

Obtaining a highly accurate model of the dynamics of a physical system becomes increasingly difficult and costly as the complexity of the system increases. Further, high-fidelity models, even when they are available, are often too computationally intensive to reason with in real time (an issue that we will explore in detail in Chapter 5). One approach to mitigate this limitation is to allow the autonomous system to either maintain and improve a model of the dynamics as it obtains data during its operation or directly adapt its decision-making using a data-driven control policy without explicitly representing the dynamics.

Methods corresponding to these two alternatives have been widely explored in the fields of adaptive control and, more recently, reinforcement learning (for a standard reference, see [61] and [62] respectively). Adaptive control made the distinction between *indirect* methods,

which conducted some form of *system identification* to model the dynamics of the system based on observations and subsequently used this model to make control decisions, and *direct* methods, which defined an automatic scheme to adjust the parameters of a control policy based on the data received, achieving desirable system adaptation. Reinforcement learning analogously distinguishes between *model-based* approaches, which seek to refine a statistical representation of the (discrete-time) transition dynamics to then inform the optimization of a decision policy, and *model-free* approaches, which aim to directly learn a value function or an efficient policy based on ongoing experience. The fundamental connection between adaptive control and reinforcement learning, often overlooked today, was actually clearly established by three of the fathers of reinforcement learning, Richard Sutton, Andrew Barto, and Ronald Williams, during its early days [63].

2.4.1 System Identification

The problem of system identification has generated an extremely rich body of literature that we cannot hope to cover here. Instead, we focus on a small number of central techniques that will be used in the upcoming chapters. We stress that the field of system identification is much broader, and direct interested readers to [64] for a modern reference and [65] for a historical review.

Bayesian Inference

Bayesian inference is an uncertainty modeling framework that uses probability calculus to maintain a probability distribution, known as a *belief*, representing a person's or a system's degree of confidence in the possible values of an unknown variable in the world. What follows is a superficial exposition of its basic functioning.

Let θ be an uncertain variable living in some measurable space Θ. We begin by specifying a *prior* probability distribution $P(\theta)$ over the space Θ that captures our initial belief before observing any evidence. We then consider a separate variable y that we can observe directly, also in a measurable space $\mathcal{Y}$. Suppose we have a probabilistic model that describes how likely different values of y are for each possible value of θ, given by the conditional probability distribution $P(y \mid \theta)$. By observing the value of variable y, we can use this information to update our belief about the unobserved variable θ of interest by applying a fundamental identity in probability calculus known as *Bayes' rule*:

$$P(\theta \mid y) = \frac{P(y \mid \theta)P(\theta)}{P(y)} \ , \tag{2.48}$$

with $P(y)$ the marginal probability of observing the given value of y aggregated over all possible values of θ (this aggregation may be in the form of either a sum or an integral, depending on the space Θ). The resulting *posterior* probability distribution can now be used as a new "prior" belief with respect to additional new evidence that may be obtained in the future. Through this process, Bayes' rule can be applied recursively as new information

is acquired, leading to an incrementally refined belief about the unknown variable θ; this iterative process is generically denoted a *Bayesian filter*.

Bayesian inference is a highly general method, which underlies a wide range of statistical learning techniques. It can be used, for example, to infer parameter values of a dynamical model from one or multiple observed state trajectories; or to maintain a belief on the evolution of the internal state of a dynamical system based on indirect observations of only some of its state variables. In the special case of a linear dynamical system with Gaussian uncertainties, this Bayesian state inference takes the form of a *Kalman filter* [66] (which also has extensions to non-linear, non-Gaussian systems), and more generally it can be implemented numerically by a sequential Monte Carlo scheme, also known as a *particle filter* [67].

Gaussian Process

A Gaussian process is a powerful probabilistic model that enables us to perform nonparametric Bayesian inference over an unknown function. A Gaussian process can be seen as the extension of the multivariate Gaussian distribution to the infinite-dimensional space of functions. Formally, it is a random field with the special property that the value of the uncertain function at any finite number of points is distributed as a multivariate normal distribution. Through this property, it is possible to fully define a Gaussian process by a mean function $\mu : \mathbb{R}^n \to \mathbb{R}$ and a positive semidefinite covariance kernel function $k : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$.

Given a *prior* Gaussian process distribution $f \sim \mathcal{GP}\big(\mu(\cdot), k(\cdot,\cdot)\big)$ and a set of observations of the value of the unknown function at finitely many points (possibly affected by Gaussian measurement noise), Bayes' rule can be performed implicitly to obtain a *posterior* Gaussian distribution on the value of the function at one or multiple query points. In particular, suppose we have N measurements $\mathbf{y} = [y_1, \dots, y_N]$, observed with independent Gaussian noise $\epsilon_i \sim \mathcal{N}(0, (\sigma_n)^2)$ at the points $X = [x_1, \dots, x_N]$, i.e. $y_i = f(x_i) + \epsilon_i$. Then the posterior probability of the function value at the set of query points X_* is given by as a multivariate normal with mean and covariance:

$$\mathbb{E}[f(X_*) \mid \mathbf{y}, X] = \mu(X_*) + K(X_*, X)(K(X, X) + (\sigma_n)^2 I)^{-1}(y - \mu(X)) \ , \tag{2.49a}$$

$$\operatorname{cov}[f(X_*) \mid X] = K(X_*, X_*) - K(X_*, X)(K(X, X) + (\sigma_n)^2 I)^{-1} K(X, X_*) \ , \tag{2.49b}$$

where $f_i(X) = f(x_i)$, $\mu_i(X) = \mu(x_i)$, and for any X, X' the matrix $K(X, X')$ is defined component-wise as $K_{ik}(X, X') = k(x_i, x'_k)$.

Since the infinite-dimensional operation takes place implicitly, Gaussian process inference can be considered a *kernel method*. Gaussian process inference is an attractive technique due to its nonparametric nature, which makes it highly flexible and expressive, and its ability to handle unknown functions in a Bayesian framework, reasoning not only about estimated values but also the degree of confidence around them. This nonparametric flexibility comes at the price of heavy computation, since the cost of necessary matrix inversion scales (roughly) cubically with the number of observed function values. In Chapter 6 we will apply Gaussian processes to reason about dynamic uncertainty in a Bayesian framework and will look into efficient computational approaches for online implementation.

2.4.2 Reinforcement Learning

The field of reinforcement learning comprises a wide variety of data-driven methods by which a system can compute approximations to the optimal value function and/or optimal policy to an optimal control problem. While we cannot hope to do justice to the rich and burgeoning literature in the field, we provide here a simple statement of its central formulation, drawing connections to optimal control theory that will be useful in the coming chapters. We direct the interested reader to [62] for a standard, recently updated reference.

Reinforcement learning is usually formulated in the discrete-time Markov decision process framework, considering the problem of maximizing the cumulative sum of rewards of a trajectory (Lagrange problem), exponentially discounted over time. In the simplest case of deterministic dynamics (2.3), maintaining notation consistent with other sections, the dynamic programming principle associated with this control problem takes the form of the discrete-time time-discounted Bellman equation [32]:

$$V(x) = \max_{u \in \mathcal{U}} L(x,u) + \gamma V\big(x + f(x,u)\delta\big) \ , \tag{2.50}$$

with $\gamma \in [0,1)$ the discount factor. Introducing uncertainty in the dynamics in the form of a probabilistic transition measure (2.9), we have the expected-value Bellman equation:

$$V(x) = \max_{u \in \mathcal{U}} L(x,u) + \gamma \int_{\mathbb{R}^n} V(z)\, dP_x(z \mid x,u) \ , \tag{2.51}$$

which corresponds to the optimal control problem of maximizing the *expected* discounted sum of rewards under (2.9).

Importantly, the *Bellman update* in the right-hand side of the two above equations induces a contraction mapping in the space of value functions (under the supremum norm), which implies that its successive application to any initial V will ultimately converge to the unique solution of (2.50) and (2.51) respectively. This enables key convergence results in a number of reinforcement learning schemes, most notably temporal-difference learning methods such as Q-learning [68].

Reinforcement learning methods are discussed in more detail in Chapter 6, where they are used in conjunction with a safe control framework, and again in Chapter 9, where they are conversely leveraged to perform safety analysis.

2.4.3 Learning and Safety

It may intuitively seem as though learning-based control and safety-preserving control are conflicting or even incompatible concepts. Indeed, a recurring concern around otherwise promising machine-learning approaches to automated decision-making is the difficulty in providing reliable assurances about their performance. Given that a system learning from data may behave unpredictably depending on the data it receives, how could such a system ever be trustworthy?

Our contention is that, while some learning approaches may exhibit brittle properties when implemented on their own (in what is commonly known as *end-to-end learning* solutions), it is possible to build intelligent systems that combine the grounding provided by data-driven reasoning with the intelligibility of model-based analysis, achieving both better performance and more reliable safety assurances than non-learning systems. At multiple points in this dissertation we will provide evidence that not only can learning systems be designed to operate with high-confidence safety assurances analogous to those we can provide in other classes of automated systems, but the learning itself can increase the robustness and resilience of model-based analysis, preventing some of the pitfalls that can result from the use of inevitably fallible models. We will further show encouraging preliminary results that suggest that the implicit exploitation of underlying structure in high-dimensional spaces may turn certain learning-based control methods into key enablers for scalable safety analysis in complex systems beyond the reach of existing computational approaches.

2.5 Cognitive Human Models

Many of the robotic and autonomous systems operating outside of controlled industrial settings will need to interact with human beings: in some cases actively assisting them with tasks, in others merely sharing the space safely. Therefore, modeling human behavior, intent, and preferences for the purpose of predicting or actively planning these interactions is instrumental for such systems to operate successfully. Over recent decades, the field of cognitive science has produced substantial advancements in our understanding of human reasoning and decision-making over. Initially spurred by the *cognitive revolution* during the second half of the 20th century [69], psychologists, linguists, neuroscientists, and computer scientists, have jointly developed and validated quantitative models of a wide range of human cognitive processes, which robotic systems can leverage to reason about their interactions with human beings.

In the 1970s, David Marr proposed a distinction between three complementary *levels of analysis* at which human cognition and the central nervous system should be studied [70]. At the highest abstraction, the *computational* level attempts to identify and formalize the problems that people are trying to mentally solve—for the cognitive process at hand, it can be thought of as addressing the question "what does the central nervous system do and why". The *algorithmic-representational* level studies the concrete strategies by these computational problems are tackled, including how the relevant inputs and outputs are represented and manipulated—it can be seen as attempting to answer the question "how does the central nervous system do what it does". Finally, the *implementational* level (sometimes also referred to as the physical or biological level) considers h encoding and transmission of information between neurons in the nervous system—the question it is concerned with is therefore "how does the central nervous system commit its strategies to physical mechanisms that can realize them".

Analysis at the computational level, then, tends to be agnostic to *how* people solve a

particular problem, a concern that falls within the algorithmic-representational level; instead, work at this level is often interested in understanding the ideal solution to the identified computational problem, such as Bayesian inference and expected utility maximization. This is often a useful starting point, to the extent that evolutionary pressure would have favored central nervous systems that roughly approximate this solution;[14] studying the differences in output between the ideal solution and actual human performance at the relevant task can in turn shed light on algorithmic-level questions.

In this dissertation, our modeling of humans can roughly be categorized as belonging to the computational level. This level is appropriate for robotic systems because it allows robots to reason about human actions as driven by human preferences or intent, producing tractable approximate predictions of human behavior in different contexts. Algorithmic-level approaches, as well as work straddling the algorithmic and the computational [71], are likely to become increasingly useful to human-centered autonomous systems in the coming years, allowing more refined and accurate reasoning about human cognition and thereby improving collaboration and interaction.

2.5.1 The Rationality Approximation

While human beings cannot possibly implement the intractable computations of an ideal rational agent, it is still useful to model their behavior as being approximately rational, in the sense of most often making decisions that are likely to result in outcomes that align well with their preferences. While there is a rich body of literature studying the numerous aspects in which this assumption is inaccurate (perhaps most notably the work by Tversky and Kahneman on heuristics and biases in human cognition [72, 73]), rationality-based models have been widely used in economics and mathematical psychology, enabling us to gain insights about how people behave in different contexts.

When considering all of these models for their use in robotics and autonomous systems, it is appropriate to keep in mind George Box's aphorism stated at the beginning of this chapter—while the models we consider will necessarily contain simplifying assumptions, what should ultimately guide our decision to use them is whether they can allow us to design better systems.

2.5.2 Luce's Choice Rule and Noisy Rationality

One particularly useful approach to reason about human decisions while acknowledging the approximate nature of our models is by having the model be probabilistic in nature. In the late 1950s, Duncan Luce formulated a *choice axiom* [74], by which the probability that a

[14] While it is extremely unlikely that any part of the human brain would run an implementation of Bayes rule and use the output to find the expected utility of each available choice, it is conceivable that those of our ancestors whose brains produced outputs far off from the "right answer" would have had worse fortune avoiding lurking predators and poisonous berries.

human would choose a particular option a in a certain pool $\mathcal{A}$ can be expressed as

$$P(a) = \frac{\nu(a)}{\sum_{\tilde{a}\in\mathcal{A}} \nu(\tilde{a})} \ , \tag{2.52}$$

for some function $\nu : \mathcal{A} \to \mathbb{R}$. This suggest that people are *likelier* to make choices that are well aligned with a particular utility model than choices that are less so—this property is often referred to as *noisy rationality*. This probabilistic uncertainty can be used to subsume the violations of this rule that we may expect due to human decisions being intrinsically irrational in some cases; more generally, it can allow us to account for the fact that our utility model itself will almost certainly be inaccurate, failing to reflect aspects of human preferences that may be relevant to certain choices.

A special case of Luce's choice rule is the *Boltzman-rational* decision model, where the function ν is exponential in the modeled utility of the given option. For example, we can model the human as choosing between multiple possible controlled state trajectories under some utility outcome functional $\mathcal{V}$.

$$P(\mathbf{u} \mid x) = \frac{e^{\mathcal{V}(x,\mathbf{u})}}{\sum_{\mathbf{u}\in\mathcal{A}_\mathbf{u}} e^{\mathcal{V}(x,\tilde{\mathbf{u}})}} \ . \tag{2.53}$$

Alternatively we can model the human as choosing one of multiple available control actions at each time step given a discrete-time dynamical system model. Instead of an outcome functional, we can then use the state-action value function Q to determine the choice probabilities.

$$P(u \mid x) = \frac{e^{Q(x,u)}}{\sum_{\mathcal{A}_u} e^{Q(x,\tilde{u})}} \ . \tag{2.54}$$

In the above models, it is most common to specify a finite set of choices. Specifying a continuum of options is also possible; in this case, the model can be used to quantify the probability density function over human choices. However, evaluating the denominator of (2.53) and (2.54), called the *partition function*, is often challenging in practice. We will be making use of noisy rationality models to reason about human decisions in Chapters 7 and 8, where we will also delve into how robotic systems may reason about the accuracy of these models in light of ongoing observations of human behavior.

2.5.3 Inverse Optimal Control

The rationality-based models discussed so far rely on some definition of utility or preferences that humans seek to realize through their decisions. A natural approach to arrive at such a representation is by observing human behavior and seeking to find a utility function that explains it.

This problem was initially posed by Rudolf Kalman in 1960 as *inverse optimal control* [75]: if we are presented with the behavior of an agent or controller, can we find the cost function for which this behavior is optimal? The same problem has received renewed attention in

the last two decades, sometimes "rebranded" as *inverse reinforcement learning* [76]. This alternative name is arguably less accurate, since the goal continues to be finding the reward function for an agent that is already executing its optimal behavior, rather than one that is in the process of *learning* an optimal behavior from experience (also an extremely interesting problem worth of research, which will unfortunately need to find a different name, since the most natural one is now taken).

The problem is in general ill-defined, since there are typically infinitely many utility functions that would make any given control policy optimal, and some of the solutions may be completely uninformative (for example any constant utility function would be indifferent to the system's behavior and therefore make *all* possible policies optimal). Instead, we may impose additional conditions on the utility function to find a unique, meaningful solution. Examples of this include maximum-margin inverse optimal control [77], Bayesian inverse optimal control [78], and maximum-entropy inverse optimal control [79]. The latter two take similar approaches based on Boltzmann-rational decision-making models to reach a point estimate or a probability distribution representing the utility function behind the observed behavior. In Chapter 7 we will build on these techniques and extend their use to explicitly reason in real time about their ability to accurately explain the observed human behavior.

2.5.4 Theory of Mind

An important consideration when robotic systems operate in close proximity to human beings is how human beings will reason about the robot's intent. By actively interacting with their environment, robots tend to be perceived by people as agents with purpose-driven behavior, which will in turn affect the way in which they expect to interact with these systems. Human beings are social in nature and therefore competent at reasoning about the internal goals and beliefs of others. This ability to represent others' mental states is what is commonly referred to in the psychology literature as *theory of mind*.[15] Recent work in robotics and neighboring fields has shown evidence that people indeed think of robots as being imbued with agency, goals, and beliefs, and that accounting for this can enable robotic systems to more effectively collaborate and interact with human beings, for example by generating *legible* motion that facilitates human inferences about its goals [83], or by choosing task plans that make it easy for a human collaborator to unambiguously predict the remaining sequence of actions [84]. The scientific understanding of human theory of mind and the existing evidence of its usefulness in robotics will serve as a basis for our proposed use of dynamic game theory to reason about strategic interactions between robotic systems and human agents in Chapter 8.

[15] Although not directly relevant to this dissertation, there is some fascinating research in the field of developmental psychology about how human beings develop theory of mind during their very early years. We direct interested readers to the work by Wimmer, Gopnik, Meltzoff, and their collaborators [80–82]

Part I

Safety Analysis for Robotic Systems

Chapter 3

Time-Varying Reach-Avoid Games

> We cannot talk of optimal pursuit without also speaking of optimal evasion.
>
> Rufus Isaacs
> *Differential Games*, 1965

This chapter is based on the paper "Reach-Avoid Problems with Time-Varying Dynamics, Targets and Constraints" [13], written in collaboration with Mo Chen, Claire Tomlin, and Shankar Sastry.

This chapter introduces some foundational mathematical machinery that will be used throughout this dissertation. In particular, it extends the existing Hamilton-Jacobi analytical framework for time-invariant problems to problems in which the system dynamics, target set, and state constraints may all be dependent on time. The time-dependence allowed is almost unlimited in flexibility, except for mild technical conditions on the variation of the dynamics and the target and constraint sets. Crucially, by building the time-varying analysis into the backward-time propagation of dynamic programming computations, the newly introduced Hamilton-Jacobi equation for time-varying problems can be solved with the same computational complexity as the traditional solution to an analogous time-invariant problem.

Dynamic reach-avoid games, which were briefly introduced in Chapter 2, have received growing interest in recent years and have many important applications in engineering problems, especially concerning the control of strategic or safety-critical systems: in many scenarios, we may want to compute a control strategy for our autonomous system that will ensure it reaches a desired region in the state space while respecting a set of constraints, often under uncertain dynamics or in the presence of an unknown disturbance or adversary. As we will see in Chapter 4, an important application of this problem is autonomous vehicle

trajectory planning, especially in large-scale traffic where coordination is possible in principle but algorithmically challenging in practice.

Finding the optimal solution to the two-player zero-sum reach-avoid differential game typically involves determining the set of states from which the *attacker* can successfully drive the system to a desired target set, while keeping the state inside a specified state constraint set at all times, in spite of the opposing actions of the *defender*. This set is commonly referred to as the *reach-avoid set* (sometimes the *capture basin*) of the target under the constraints, and it also corresponds to the *victory domain* (or *winning set*) of the attacker. In addition to this set, it is useful and desirable to characterize the optimal control strategies for both players. In the context of this dissertation, the optimal attacker strategies will inform the design of our autonomous system controller; on the other hand, studying the optimal disturbance strategies can provide useful insights about weaknesses or liabilities in the current system design.

The mathematical formulation introduced in this chapter will be instrumental in the development of decision-making schemes in Chapter 4, and is also highly relevant to the safety analysis in Chapters 5–7.

Related Work

In the absence of state constraints, reachability problems involving possibly time-varying target sets can be posed as a maximum (minimum) cost game where the players try to optimize the pointwise minimum over time of some metric to the target. In this case, the backwards reachable set can be obtained by finding the viscosity solution to the corresponding Hamilton-Jacobi-Isaacs (HJI) equation in the form of a variational inequality: this value function captures the minimum distance to the target that will be achieved by the optimal trajectory starting at each point, so the reach-avoid set is characterized by the region of the state space where this minimum future distance is equal to or less than zero. Maximum cost control problems were studied in detail in [85], and extended to the two-player setting in [48]. While computationally intensive, Hamilton-Jacobi approaches are practically appealing nowadays due to the availability of modern numerical tools such as [52, 53, 86], which allow solving the associated equations for problems with low dimensionality.

If the game is played under state constraints, then the value function generally becomes discontinuous [87], which leads to numerical issues. In the case of systems with time-invariant dynamics, targets and constraints, the approach in [88] characterizes the reach-avoid set through an auxiliary value function that solves a modified Hamilton-Jacobi variational inequality. Although the new value function no longer captures the minimum distance from a trajectory to the target, the reach-avoid set is still given by the value function's subzero region. This allows to effectively turn a constrained final cost problem into an unconstrained problem with a maximum cost.

For problems with time-varying dynamics, targets and constraints, the approach proposed in [89] as an extension of [88] requires augmenting the state space with an additional dimension accounting for time; one can then transform time-dependence into state-dependence

and apply the above described methods to solve the fixed problem in the space-time state space. Unfortunately, this approach presents a significant drawback, since the complexity of numerical computations is exponential in the problem dimensionality.

Contribution

The main contribution of this chapter is an extension of the Hamilton-Jacobi reach-avoid formulation to the case where the target set, the state constraints, and dynamics are allowed to be time-varying, enabling computation of the reach-avoid set at no significant additional cost relative to the time-invariant case. To this end, we formulate a new *double-obstacle* Hamilton-Jacobi-Isaacs variational inequality, and prove that the zero sublevel set of its viscosity solution characterizes the desired reach-avoid set.

We also provide a numerical scheme based on [51, 52] and implementation based on [53] to solve the variational inequality and verify the numerical solution using a simple example. We finish by showing that our method vastly outperforms techniques requiring state augmentation.

Other authors have recently studied Hamilton-Jacobi equations with a double obstacle in the context of games with imperfect information [90] and stochastic games with impulse controls [91]. To our knowledge, however, the work presented in this chapter constitutes the first analysis of double-obstacle Hamilton-Jacobi equations in the context of reachability problems.

3.1 Time-Varying Reach-Avoid Games

Consider a dynamical system controlled over a time horizon $[0, T]$ by two agents with opposing objectives. As in Section 2.3.5, we will adopt the convention that one of them is our system controller and the other an adversarial disturbance:

$$\dot{x} = f(x, u, d, t) \ , \tag{3.1}$$

with f being bounded, uniformly continuous, and additionally Lipschitz-continuous in x.

Recall that a reach-avoid game is a *game of kind* in which the objective of our controller (otherwise known as the *attacker* in this context), is to reach a target set $\mathcal{T} \subseteq \mathbb{R}^n$ while remaining inside a constraint set $\mathcal{K} \subseteq \mathbb{R}^n$; the worst-case disturbance (here in the role of the *defender*) will act on the system to prevent the controller from succeeding whenever possible, either by keeping the state away from the target or by driving it into a constraint violation.

In this chapter, we are interested in the case where the target and the constraint may not be static subsets of $\mathbb{R}^n$, but may instead evolve over time. In fact, we would like to make the evolution as general as possible, so that the regions of interest may not only move over time, but also change shape and size, split into multiple subregions, or even appear and disappear instantly. Similar to how our formulation of safety problems and reach-avoid games presented in Chapter 2 considered arbitrary target and constraint sets (with only a

closedness convention adopted for unambiguous level set interpretation), our aim here is to develop an analysis that additionally allows arbitrary *behavior* of these sets over time.

Also similar to time-invariant reach-avoid problems, our formulation assumes that players have perfect environment information: this implies that players know ahead of time how the target, constraints, and system dynamics will evolve during the game. This formulation may seem overly restrictive at first, but as we will see in future sections it enables the treatment of a variety of interesting engineering problems. It is worth noting that an object with uncertain motion can always be represented, using worst-case analysis, as a growing obstacle (through its maximal forward-reachable set at each future time) or a shrinking target (minimal forward-reachable set).

By a slight abuse of notation, we define the set-valued maps $\mathcal{T}, \mathcal{K} : [0,T] \rightrightarrows \mathbb{R}^n$ which respectively assign a target set $\mathcal{T}(t) \subseteq \mathbb{R}^n$ and a constraint set $\mathcal{K}(t) \subseteq \mathbb{R}^n$ to each time $t \in [0,T]$. As usual we will adopt the convention that $\mathcal{T}(t)$ and $\mathcal{K}(t)$ are closed for all $t \in [0,T]$. Our single requirement for these set-valued maps is that they are *upper hemicontinuous*.[1] This requirement is not very restrictive, and allows a wide variety of target and constraint behaviors including discontinuous changes like intermittently appearing and disappearing. In fact, the purpose of this assumption is to extend the closedness of the target and the constraint to the space-time domain. We can construct the space-time sets

$$\mathbb{T} := \bigcup_{t\in[0,T]} \mathcal{T}(t) \times \{t\} \ , \qquad \mathbb{K} := \bigcup_{t\in[0,T]} \mathcal{K}(t) \times \{t\} \ , \tag{3.2}$$

which are then closed subsets of $\mathbb{R}^n \times [0,T]$ by the following standard result.

Lemma 3.1 (Closed Graph Theorem)**.** *Let $\mathcal{M} : [0,T] \rightrightarrows \mathcal{X} \subseteq \mathbb{R}^n$ be an upper hemicontinuous set-valued map with $\mathcal{M}_t = \mathcal{M}_t$ closed in $\mathbb{R}^n$ for all $t \in [0,T]$. Then the set $\mathbb{M} = \bigcup_{t\in[0,T]} \mathcal{M}_t \times \{t\}$ is closed in $\mathbb{R}^n \times [0,T]$. If $\mathcal{X}$ is compact, the converse is also true.*

The closed sets $\mathbb{T}$ and $\mathbb{K}$ can then be implicitly characterized as the subzero regions of two Lipschitz-continuous functions $l : \mathbb{R}^n \times [0,T] \to \mathbb{R}$ and $g : \mathbb{R}^n \times [0,T] \to \mathbb{R}$ respectively:

$$\begin{aligned} x \in \mathcal{T}(t) &\iff (x,t) \in \mathbb{T} \iff l(x,t) \leq 0 \ , \\ x \in \mathcal{K}(t) &\iff (x,t) \in \mathbb{K} \iff g(x,t) \leq 0 \ . \end{aligned} \tag{3.3}$$

These functions always exist, since we can simply choose the signed distance functions $l(x,t) = s_{\mathbb{T}}(x,t)$ and $g(x,t) = s_{\mathbb{K}}(x,t)$, which are Lipschitz by construction. Note that, contrary to the convention in safety problems, reach-avoid problems define g to be *negative* inside $\mathcal{K}$; we thus follow the appropriate sign convention in this chapter.

[1] A set-valued map $\mathcal{M} : [0,T] \rightrightarrows \mathbb{R}^n$ is upper hemicontinuous if for any open neighborhood $\mathcal{X}$ of $\mathcal{M}_t$ there is an open neighborhood (t_1, t_2) of t such that $\mathcal{M}_\tau \subseteq \mathcal{X}$ $\forall \tau \in (t_1, t_2)$.

3.1.1 Value of the Game

The property of the state evolution that the controller and the disturbance are contending over is the *reach-avoid condition*. For any given initial conditions (x, t) and input signals $\mathbf{u}, \mathbf{d}$, this logical condition is true if the resulting state trajectory finds itself in the target set at some future time without previously violating the constraints, that is:

$$\exists \tau \in [t, T] \mid \quad \mathbf{x}_{x,t}^{\mathbf{u},\mathbf{d}}(\tau) \in \mathcal{T}(\tau) \;\; \wedge \;\; \forall s \in [t, \tau], \mathbf{x}_{x,t}^{\mathbf{u},\mathbf{d}}(\tau) \in \mathcal{K}(s) \;\; . \tag{3.4}$$

In the reach-avoid differential game, the controller will try to make (3.4) true from each given (x, t), whereas the disturbance will attempt to make it false. This game of kind can be encoded by a game of degree with the following real-valued functional:

$$\mathcal{V}\big(x, t, \mathbf{u}, \mathbf{d}\big) := \min_{\tau \in [t,T]} \max \Big\{ l\big(\mathbf{x}_{x,t}^{\mathbf{u},\mathbf{d}}(\tau), \tau\big), \; \max_{s \in [t,\tau]} g\big(\mathbf{x}_{x,t}^{\mathbf{u},\mathbf{d}}(s), s\big) \Big\} \;\; . \tag{3.5}$$

The following result follows directly from (3.4) and (3.5) by construction, and is straightforward to verify by inspection.

Proposition 3.1 (Reach-avoid game of degree encoding)**.** *A system trajectory* $\mathbf{x}_{x,t}^{\mathbf{u},\mathbf{d}}$ *satisfies the reach-avoid condition* (3.4) *if and only if the associated outcome* $\mathcal{V}(x, t, \mathbf{u}, \mathbf{d})$ *is non-positive.*

$$\exists \tau \in [t, T] \mid \mathbf{x}_{x,t}^{\mathbf{u},\mathbf{d}}(\tau) \in \mathcal{T}(\tau) \;\wedge\; \forall s \in [t, \tau], \mathbf{x}_{x,t}^{\mathbf{u},\mathbf{d}}(\tau) \in \mathcal{K}(s) \iff \mathcal{V}\big(x, t, \mathbf{u}, \mathbf{d}\big) \leq 0 \;\; . \tag{3.6}$$

Exploiting Proposition 3.1 through the level set approach introduced in Chapter 2, we can analyze a problem in which the controller and the disturbance compete over the continuous value of $\mathcal{V}$, and then use its sign to determine whether or not the controller succeeds in satisfying the reach-avoid condition.

In order to have a well-defined notion of value, we now need to specify the information structure of the game. As in Chapter 2, we will be allowing the disturbance to use non-anticipative strategies $\boldsymbol{\delta} : \mathbb{U} \to \mathbb{D}$ (as per Definition 2.3 in Section 2.2), thus giving it the instantaneous informational advantage relative the controller. As we saw, this effectively means that the controller has access to state feedback information, whereas the disturbance can additionally observe the controller's instantaneous input before introducing its own. This matches our conceptualization of robust decision-making under model uncertainty: while feedback information allows accounting for the current state of the system, the imminent effect of a control input may not be exactly anticipated at each instant, and could happen to be the worst possible for any given choice.

Since in this case, we have the controller trying to make the outcome of the game $\mathcal{V}$ as low as possible, this information structure leads to the *upper value* of the game:

$$V^+(x, t) := \sup_{\boldsymbol{\delta} \in \mathfrak{D}_t} \inf_{\mathbf{u} \in \mathbb{U}_t^T} \mathcal{V}\big(x, t, \mathbf{u}, \boldsymbol{\delta}[\mathbf{u}]\big) \;\; . \tag{3.7}$$

Remark 3.1 (Alternative information structures)**.** *The lower value can be defined similarly by giving the controller the instantaneous informational advantage instead. This would rarely be meaningful from an engineering standpoint, since disturbances are not generally measurable ahead of time. Instead, in some cases we may wish to retain the information structure but reverse the optimization goals of the controller and the disturbance, namely whenever the specified goal is to prevent a certain reach-avoid condition from being met. Finally, there will be cases where the upper and lower values are equal to each other and there is therefore no instantaneous informational advantage to be exploited. In this chapter, we focus our analysis on the upper value of the game as defined by* (3.7)*, noting that the analysis for the other cases is analogous.*

Definition 3.1 (Space-time reach-avoid set)**.** *We say that initial conditions (x,t) are in the space-time reach-avoid set $\mathbb{RA} \subseteq \mathbb{R}^n \times [0,T]$ when the system trajectory $\mathbf{x}_{x,t}^{\mathbf{u},\mathbf{d}}$ resulting from both players acting optimally for a given information pattern reaches $\mathcal{T}(t)$ at some time $\tau \in [t,T]$ while remaining in $\mathcal{K}(s)$ at all intermediate times $s \in [t,\tau]$. In particular, when the disturbance uses non-anticipative strategies, we have the robust space-time reach-avoid set*

$$\begin{aligned}\overline{\mathbb{RA}} := \{(x,t) \in \mathbb{R}^n \times [0,T] : \exists \mathbf{u} \in \mathbb{U}_t^T, \forall \boldsymbol{\delta} \in \mathfrak{D}_t, \\ \exists \tau \in [t,T], \mathbf{x}_{x,t}^{\mathbf{u},\boldsymbol{\delta}[\mathbf{u}]}(\tau) \in \mathcal{T}(\tau) \ \wedge \ \forall s \in [t,\tau], \mathbf{x}_{x,t}^{\mathbf{u},\boldsymbol{\delta}[\mathbf{u}]}(s) \in \mathcal{K}(s)\},\end{aligned} \tag{3.8}$$

Combining Definition 3.1 with Proposition 3.1, we have the following important result.

Proposition 3.2. *The robust space-time reach-avoid set of the space-time target $\mathbb{T}$ under space-time constraints $\mathbb{K}$ is equal to the zero sublevel set of the upper value function V^+:*

$$\overline{\mathbb{RA}} = \{(x,t) \in \mathbb{R}^n \times [0,T] : V^+(x,t) \leq 0\} \ . \tag{3.9}$$

Proof. Let us first show that $(x,t) \in \overline{\mathbb{RA}} \Rightarrow V^+(x,t) \leq 0$. If (x,t) is in the robust space-time reach-avoid set, this means that for all possible non-anticipative strategies $\boldsymbol{\delta} \in \mathfrak{D}_t$ available to the disturbance there is some control signal $\mathbf{u} \in \mathbb{U}_t^T$ for which the state trajectory $\mathbf{x}_{x,t}^{\mathbf{u},\boldsymbol{\delta}[\mathbf{u}]}$ will eventually be in $\mathcal{T}(\tau)$ for some $\tau \in [t,T]$ while having remained in $\mathcal{K}(s)$ at all intermediate times $s \in [t,\tau]$. By Proposition 3.1, this means that the outcome $\mathcal{V}(x,t,\mathbf{u},\boldsymbol{\delta}[\mathbf{u}])$ can always be made non-positive by the controller for each choice of $\boldsymbol{\delta} \in \mathfrak{D}_t$ made by the disturbance, which, applying the definition of the upper value in (3.7), implies that $V^+(x,t) \leq 0$.

For the opposite direction, suppose that $V^+(x,t) \leq 0$. This means that regardless of the non-anticipative strategy $\boldsymbol{\delta} \in \mathfrak{D}_t$ chosen by the disturbance there will exist some control signal $\mathbf{u} \in \mathbb{U}_t^T$ for which the outcome $\mathcal{V}(x,t,\mathbf{u},\boldsymbol{\delta}[\mathbf{u}])$ is non-positive. By Proposition 3.1 this implies that $\mathbf{x}_{x,t}^{\mathbf{u},\boldsymbol{\delta}[\mathbf{u}]}$ satisfies the reach-avoid condition. Thus, by Definition 3.1, $(x,t) \in \overline{\mathbb{RA}}$. □

This proposition means that the value function encodes the optimal outcome of the reach-avoid game. Put in Isaacs' terms, the value of the *game of degree* at initial conditions (x,t) is enough to determine which player will win the *game of kind* from these conditions. Therefore,

if we are able to compute the value function V^+, we can use this to provide robust guarantees on the controller's ability to reach the possibly time-varying target $\mathcal{T}(t)$ while remaining in the possibly time-varying constraints $\mathcal{K}(t)$, under possibly time-varying dynamics (3.1). Further, we will see that the optimal feedback control policy implicitly obtained in the computation of V^+ has prescriptive value in enforcing the reach-avoid guarantee.

Analogously to the time-varying target and constraint sets, we can define the time-varying reach-avoid set as a set-valued map $\mathcal{RA} : [0, T] \rightrightarrows \mathbb{R}^n$ such that

$$x \in \overline{\mathcal{RA}}_t \iff (x, t) \in \overline{\mathbb{RA}} \iff V^+(x, t) \leq 0 \ . \tag{3.10}$$

3.2 The Double-Obstacle Isaacs Equation

As we saw in Chapter 2, the value function for minimum-payoff games (which can be used to encode safety games) can be characterized as the unique viscosity solution to a variational inequality [92], which has commonly been referred to as a Hamilton-Jacobi equation "with an obstacle", due to the presence of a term that effectively saturates the propagation of the value function. In this section, we extend the results for minimum payoff problems to the category of problems with an outcome in the form of (3.5) (useful for encoding reach-avoid problems), and we show that the value function is the viscosity solution to Hamilton-Jacobi equation with two "obstacles", one of them saturating its propagation from above and the other from below.

3.2.1 Dynamic Programming Principle

We begin by stating the particular form of Bellman's principle of optimality [32] (or strictly speaking Isaacs' tenet of transition [36]) for the problem at hand, noting that it is quite different from the more typical relation in (2.17).

Lemma 3.2 (Dynamic Programming Principle)**.** *Let V^+ be the upper value defined in* (3.7)*. For any $t \in [0, T]$ and $\delta \in [0, T - t]$, the following holds.*

$$V^+(x,t) = \sup_{\boldsymbol{\delta} \in \mathfrak{D}_t^{t+\delta}} \inf_{\mathbf{u} \in \mathbb{U}_t^{t+\delta}} \min \Bigg\{ \min_{\tau \in [t, t+\delta]} \max \Big\{ l\big(\mathbf{x}_{x,t}^{\mathbf{u},\boldsymbol{\delta}[\mathbf{u}]}(\tau), \tau\big), \max_{s \in [t,\tau]} g\big(\mathbf{x}_{x,t}^{\mathbf{u},\boldsymbol{\delta}[\mathbf{u}]}(s), s\big) \Big\}, \tag{3.11}$$
$$\max \Big\{ V^+\big(\mathbf{x}_{x,t}^{\mathbf{u},\boldsymbol{\delta}[\mathbf{u}]}(t+\delta), t+\delta\big), \max_{\tau \in [t, t+\delta]} g\big(\mathbf{x}_{x,t}^{\mathbf{u},\boldsymbol{\delta}[\mathbf{u}]}(\tau), \tau\big) \Big\} \Bigg\} \ .$$

Proof. By the principle of optimality (tenet of transition), an optimal trajectory $\mathbf{x}_{x,t}^{\mathbf{u},\boldsymbol{\delta}[\mathbf{u}]}$ on $[t, T]$ must have the property that its restriction to $[t+\delta, T]$ for $\delta \in [0, T-t]$ is also an optimal trajectory for the remaining duration of the game. Specifically, if $(\mathbf{u}, \boldsymbol{\delta})$ is an optimal pair of

strategies on $[t,T]$, their restriction to $[t+\delta,T]$ also forms an optimal pair.[2] This allows us to appropriately decompose the optimization over $(\mathbf{u},\boldsymbol{\delta})$ in the last of the following operations:

$$
\begin{aligned}
V^+(x,t) &:= \sup_{\boldsymbol{\delta}\in\mathfrak{D}_t}\inf_{\mathbf{u}\in\mathbb{U}_t^T}\min_{\tau\in[t,T]}\max\Big\{l\big(\mathbf{x}_{x,t}^{\mathbf{u},\boldsymbol{\delta}[\mathbf{u}]}(\tau),\tau\big),\ \max_{s\in[t,\tau]} g\big(\mathbf{x}_{x,t}^{\mathbf{u},\boldsymbol{\delta}[\mathbf{u}]}(s),s\big)\Big\}\\
&= \sup_{\boldsymbol{\delta}\in\mathfrak{D}_t}\inf_{\mathbf{u}\in\mathbb{U}_t^T}\min\Bigg\{\min_{\tau\in[t,t+\delta]}\max\Big\{l\big(\mathbf{x}_{x,t}^{\mathbf{u},\boldsymbol{\delta}[\mathbf{u}]}(\tau),\tau\big),\ \max_{s\in[t,\tau]} g\big(\mathbf{x}_{x,t}^{\mathbf{u},\boldsymbol{\delta}[\mathbf{u}]}(s),s\big)\Big\},\\
&\qquad\qquad \min_{\tau\in[t+\delta,T]}\max\Big\{l\big(\mathbf{x}_{x,t}^{\mathbf{u},\boldsymbol{\delta}[\mathbf{u}]}(\tau),\tau\big),\ \max_{s\in[t,\tau]} g\big(\mathbf{x}_{x,t}^{\mathbf{u},\boldsymbol{\delta}[\mathbf{u}]}(s),s\big)\Big\}\Bigg\}\\
&= \sup_{\boldsymbol{\delta}\in\mathfrak{D}_t}\inf_{\mathbf{u}\in\mathbb{U}_t^T}\min\Bigg\{\min_{\tau\in[t,t+\delta]}\max\Big\{l\big(\mathbf{x}_{x,t}^{\mathbf{u},\boldsymbol{\delta}[\mathbf{u}]}(\tau),\tau\big),\ \max_{s\in[t,\tau]} g\big(\mathbf{x}_{x,t}^{\mathbf{u},\boldsymbol{\delta}[\mathbf{u}]}(s),s\big)\Big\},\\
&\qquad\qquad \min_{\tau\in[t+\delta,T]}\max\Big\{l\big(\mathbf{x}_{x,t}^{\mathbf{u},\boldsymbol{\delta}[\mathbf{u}]}(\tau),\tau\big),\ \max_{s\in[t,t+\delta]} g\big(\mathbf{x}_{x,t}^{\mathbf{u},\boldsymbol{\delta}[\mathbf{u}]}(s),s\big),\\
&\qquad\qquad\qquad \max_{s\in[t+\delta,\tau]} g\big(\mathbf{x}_{x,t}^{\mathbf{u},\boldsymbol{\delta}[\mathbf{u}]}(s),s\big)\Big\}\Bigg\}\\
&= \sup_{\boldsymbol{\delta}\in\mathfrak{D}_t}\inf_{\mathbf{u}\in\mathbb{U}_t^T}\min\Bigg\{\min_{\tau\in[t,t+\delta]}\max\Big\{l\big(\mathbf{x}_{x,t}^{\mathbf{u},\boldsymbol{\delta}[\mathbf{u}]}(\tau),\tau\big),\ \max_{s\in[t,\tau]} g\big(\mathbf{x}_{x,t}^{\mathbf{u},\boldsymbol{\delta}[\mathbf{u}]}(s),s\big)\Big\},\\
&\qquad\qquad \max\Bigg[\max_{s\in[t,t+\delta]} g\big(\mathbf{x}_{x,t}^{\mathbf{u},\boldsymbol{\delta}[\mathbf{u}]}(s),s\big),\\
&\qquad\qquad\qquad \min_{\tau\in[t+\delta,T]}\max\Big\{l\big(\mathbf{x}_{x,t}^{\mathbf{u},\boldsymbol{\delta}[\mathbf{u}]}(\tau),\tau\big),\ \max_{s\in[t+\delta,\tau]} g\big(\mathbf{x}_{x,t}^{\mathbf{u},\boldsymbol{\delta}[\mathbf{u}]}(s),s\big)\Big\}\Bigg]\Bigg\}\\
&= \sup_{\boldsymbol{\delta}\in\mathfrak{D}_t^{t+\delta}}\inf_{\mathbf{u}\in\mathbb{U}_t^{t+\delta}}\min\Bigg\{\min_{\tau\in[t,t+\delta]}\max\Big\{l\big(\mathbf{x}_{x,t}^{\mathbf{u},\boldsymbol{\delta}[\mathbf{u}]}(\tau),\tau\big),\ \max_{s\in[t,\tau]} g\big(\mathbf{x}_{x,t}^{\mathbf{u},\boldsymbol{\delta}[\mathbf{u}]}(s),s\big)\Big\},\\
&\qquad\qquad \max\Bigg[\max_{\tau\in[t,t+\delta]} g\big(\mathbf{x}_{x,t}^{\mathbf{u},\boldsymbol{\delta}[\mathbf{u}]}(\tau),\tau\big),\ \sup_{\tilde{\boldsymbol{\delta}}\in\mathfrak{D}_{t+\delta}}\inf_{\tilde{\mathbf{u}}\in\mathbb{U}_{t+\delta}}\min_{\tau\in[t+\delta,T]}\\
&\qquad\qquad\qquad \max\Big\{l\big(\mathbf{x}_{\tilde{x},t+\delta}^{\tilde{\mathbf{u}},\tilde{\boldsymbol{\delta}}[\tilde{\mathbf{u}}]}(\tau),\tau\big),\ \max_{s\in[t+\delta,\tau]} g\big(\mathbf{x}_{\tilde{x},t+\delta}^{\tilde{\mathbf{u}},\tilde{\boldsymbol{\delta}}[\tilde{\mathbf{u}}]}(s),s\big)\Big\}\Bigg]\Bigg\}
\end{aligned}
$$

with $\tilde{x} := \mathbf{x}_{x,t}^{\mathbf{u},\boldsymbol{\delta}[\mathbf{u}]}(t+\delta)$. Applying the definition of upper value (3.7) to the optimization over $(\tilde{\mathbf{u}},\tilde{\boldsymbol{\delta}})$ on $[t+\delta,T]$ results in the dynamic programming equality stated in the lemma. □

[2]Note that for any non-anticipative map $\boldsymbol{\delta}:\mathbb{U}_t^T\to\mathbb{D}_t^T$ and measurable prefix signal $\mathbf{u}_1:[t,t+\delta]\to\mathcal{U}$ there always exists a corresponding non-anticipative map $\boldsymbol{\delta}_2:\mathbb{U}_{t+\delta}\to\mathbb{D}_{t+\delta}$ such that for each measurable $\mathbf{u}\in\mathbb{U}_t^T \mid \mathbf{u}(\tau)=\mathbf{u}_1(\tau),\forall\tau\in[t,t+\delta]$ it holds that $\boldsymbol{\delta}_2[\mathbf{u}_2](s)=\boldsymbol{\delta}[\mathbf{u}](s),\forall s\in[t+\delta,T]$, where $\mathbf{u}_2:[t+\delta,T]\to\mathcal{U}$ is the measurable signal defined as $\mathbf{u}_2(s):=\mathbf{u}(s),\forall\tau\in[t+\delta,T]$. Therefore the restriction of a pair of strategies $(\mathbf{u},\boldsymbol{\delta})$ to a time interval $[t+\delta,T]$ is well-defined.

3.2.2 Hamilton-Jacobi-Isaacs Equation

As in the optimal control and differential game settings presented in Chapter 2, we will make use of a Hamiltonian to translate the competition between $\mathbf{u}$ and $\boldsymbol{\delta}$ into an infinite family of instantaneous zero-sum games between u and d for each (x,t). We introduce the *upper Hamiltonian* H^+ of the reach-avoid game as

$$H^+(x,p,t) = \min_{u\in\mathcal{U}} \max_{d\in\mathcal{D}} \; p\cdot f(x,u,d,t)\cdot \quad . \tag{3.12}$$

Note that the order of optimization in the upper Hamiltonian is such that the controller "plays first" in each of these games, i.e. the disturbance's input can be a function of the controller's; this is consistent with the instantaneous informational advantage given to the disturbance. Also note that, given that f is continuous, H^+ is a continuous function by the *maximum theorem*.

Before stating and proving our main result, we present here a special case of important lemma from the seminal work by Evans and Souganidis [34], which will allow us to connect the instantaneous minimax in the Hamiltonian to the non-anticipative strategies extending over time.

Lemma 3.3 (Hamiltonian bound integration)**.** *Let $\psi : \mathbb{R}^n \times [0,T] \to \mathbb{R}$ be continuously differentiable.*

(a) Suppose ψ satisfies

$$\partial_t\psi(x,t) + H^+\left(x, \nabla_x\psi(x,t), t\right) \leq -\theta \quad , \tag{3.13}$$

for some $\theta > 0$. Then there exists a sufficiently small $\delta > 0$ such that for all non-anticipative strategies $\boldsymbol{\delta} \in \mathfrak{D}_t$ there exists an input signal $\mathbf{u} \in \mathbb{U}_t$ for which

$$\psi\big(\mathbf{x}_{x,t}^{\mathbf{u},\boldsymbol{\delta}[\mathbf{u}]}(t+\delta), t+\delta\big) - \psi(x,t) \leq -\frac{\theta}{2}\delta \quad . \tag{3.14}$$

(b) Suppose ψ satisfies

$$\partial_t\psi(x,t) + H^+\left(x, \nabla_x\psi(x,t), t\right) \geq \theta \quad , \tag{3.15}$$

for some $\theta > 0$. Then there exists a sufficiently small $\delta > 0$ such that there is a non-anticipative strategy $\boldsymbol{\delta} \in \mathfrak{D}_t$ for which, regardless of the input signal $\mathbf{u} \in \mathbb{U}_t^T$,

$$\psi\big(\mathbf{x}_{x,t}^{\mathbf{u},\boldsymbol{\delta}[\mathbf{u}]}(t+\delta), t+\delta\big) - \psi(x,t) \geq \frac{\theta}{2}\delta \quad . \tag{3.16}$$

Proof. We give here an adaptation of the proof of Lemma 4.3 in [34] due to the useful intuition it provides for locally bounding the game outcome of a trajectory based on the Hamiltonian. Let

$$\Lambda(x,u,d,t) := \partial_t\psi(x,t) + \nabla_x\psi(x,t)\cdot f(x,u,d,t) \quad .$$

(a) Assumption (3.13) means that

$$\min_{u\in\mathcal{U}}\max_{d\in\mathcal{D}}\Lambda(x,u,d,t)\leq-\theta<0\ .$$

Therefore there exists some $u^*\in\mathcal{U}$ for which

$$\max_{d\in\mathcal{D}}\Lambda(x,u^*,d,t)\leq-\theta\ .$$

Since Λ is continuous by construction, and due to the continuity of state trajectories, we have that for a sufficiently small $\delta>0$ and for any signals $\mathbf{u},\mathbf{d}$,

$$\max_{d\in\mathcal{D}}\Lambda\big(\mathbf{x}_{x,t}^{\mathbf{u},\mathbf{d}}(\tau),u^*,d,\tau\big)\leq-\frac{\theta}{2}\ ,\qquad\forall\tau\in[t,t+\delta]\ .$$

Therefore, choosing the measurable signal $\mathbf{u}(\cdot)\equiv u^*$ and any $\boldsymbol{\delta}\in\mathfrak{D}_t$,

$$\partial_t\psi\big(\mathbf{x}_{x,t}^{\mathbf{u},\mathbf{d}}(\tau),\tau\big)+\nabla_x\psi\big(\mathbf{x}_{x,t}^{\mathbf{u},\mathbf{d}}(\tau),\tau\big)\cdot f\big(\mathbf{x}_{x,t}^{\mathbf{u},\mathbf{d}}(\tau),\mathbf{u}(\tau),\boldsymbol{\delta}[\mathbf{u}](\tau),\tau\big)\leq-\frac{\theta}{2}\ ,\qquad\forall\tau\in[t,t+\delta]\ .$$

Noticing that the left-hand side of the inequality is the total particle derivative of ψ along the flow f, we can integrate on the interval $[t,t+\delta]$ to obtain the upper bound (3.14), enforceable by the controller.

(b) The reasoning here is slightly more involved than in the first part, since we need to specify a well-defined adaptation $\mathcal{U}\to\mathcal{D}$, which we will then use to construct a non-anticipative strategy $\mathbb{U}_t^T\to\mathbb{D}_t^T$ for the disturbance. Assumption (3.15) means that

$$\min_{u\in\mathcal{U}}\max_{d\in\mathcal{D}}\Lambda(x,u,d,t)\geq\theta>0\ .$$

Therefore for each $u\in\mathcal{U}$ there exists some $d^*\in\mathcal{D}$ for which

$$\max_{d\in\mathcal{D}}\Lambda(x,u,d^*,t)\geq\theta\ .$$

Since Λ is continuous, there exists an open ball $B(u,r)$ with $r>0$ such that

$$\Lambda(x,\tilde{u},d^*,t)\geq\frac{3\theta}{4}\ ,\qquad\forall\tilde{u}\in B(u,r)\ .$$

Since $\mathcal{U}$ is compact, we can find a finite cover of open balls defined by centers $u_1,\dots,u_m\in\mathcal{U}$ and their respective radii $r_1,\dots,r_m>0$, satisfying $\mathcal{U}\subset B(u_1,r_1)\cup\ldots\cup B(u_m,r_m)$ and chosen in such a way that

$$\Lambda(x,\tilde{u},d_i,t)\geq\frac{3\theta}{4}\ ,\qquad\forall\tilde{u}\in B(u_i,r_i)\ ,$$

for $d_1, \ldots, d_m \in \mathcal{D}$. We can then construct the mapping $\phi : \mathcal{U} \to \mathcal{D}$ so that

$$\phi(u) := d_i \quad \text{if } u \in B(u_i, r_i) \setminus \bigcup_{j=1}^{i-1} B(u_j, r_j) \quad (i = 1, \ldots, m) \ .$$

With this, we have a well-defined instantaneous strategy ϕ for the disturbance, with the property that

$$\max_{d \in \mathcal{D}} \Lambda(x, u, \phi(u), t) \geq \frac{3\theta}{4} \ ,$$

With this in place, the remainder of the reasoning is similar to the first case. Due to the continuity of state trajectories, we have that for a sufficiently small $\delta > 0$ and for any signals $\mathbf{u}, \mathbf{d}$,

$$\max_{d \in \mathcal{D}} \Lambda\big(\mathbf{x}_{x,t}^{\mathbf{u},\mathbf{d}}(\tau), u, \phi(u), \tau\big) \geq \frac{\theta}{2} \ , \qquad \forall \tau \in [t, t+\delta] \ .$$

Therefore, choosing the non-anticipative strategy $\boldsymbol{\delta}[\mathbf{u}](\tau) := \phi\big(\mathbf{u}(\tau)\big)$ and any $\mathbf{u} \in \mathbb{U}_t^T$,

$$\partial_t \psi\big(\mathbf{x}_{x,t}^{\mathbf{u},\mathbf{d}}(\tau), \tau\big) + \nabla_x \psi\big(\mathbf{x}_{x,t}^{\mathbf{u},\mathbf{d}}(\tau), \tau\big) \cdot f\big(\mathbf{x}_{x,t}^{\mathbf{u},\mathbf{d}}(\tau), \mathbf{u}(\tau), \boldsymbol{\delta}[\mathbf{u}](\tau), \tau\big) \geq \frac{\theta}{2} \ , \qquad \forall \tau \in [t, t+\delta] \ .$$

As before, the left-hand side of the inequality is the total particle derivative of ψ along the flow f, so integrating on the interval $[t, t+\delta]$ gives the lower bound (3.16), enforceable by the disturbance. □

The above lemma is useful when in proofs related to viscosity solutions, since, as we saw in Chapter 2, their definition involves Hamilton-Jacobi-type inequalities applied to continuously differentiable functions. In addition, the constructive structure of the proof is useful for reasoning about the practical implementation of control systems based on these theoretical results (since in reality it is not possible to build a physical controller that instantaneously chooses the optimal control input at every state).

The following theorem constitutes the central theoretical contribution of this chapter: it shows that the upper value V^+ is the viscosity solution of a particular variational inequality that has the form of a Hamilton-Jacobi-Isaacs equation with a double obstacle. The result is straightforward to extend to the lower value as well, and all the proofs can be analogously derived by simply assigning non-anticipative strategies to the minimizing player instead [13].

Theorem 3.1 (Double-Obstacle Isaacs Equation)**.** *Let $f : \mathbb{R}^n \times \mathcal{U} \times \mathcal{D} \times [0,T] \to \mathbb{R}^n$ be bounded, uniformly continuous, and Lipschitz-continuous in its first variable, and suppose that $l, g : \mathbb{R}^n \times [0,T] \to \mathbb{R}$ are Lipschitz continuous functions. Then the upper value function $V^+ : \mathbb{R}^n \times [0,T] \to \mathbb{R}$ of the game with outcome given by* (3.5) *is the unique viscosity solution of the variational inequality*

$$0 = \max\Big\{ \min\Big\{\partial_t V + H^+\left(x, \nabla_x V, t\right), l(x,t) - V(x,t)\Big\}, g(x,t) - V(x,t)\Big\}, \qquad \text{(3.17a)}$$

with terminal condition

$$V(x,T) = \max\big\{l(x,T), g(x,T)\big\}, \quad x \in \mathbb{R}^n. \tag{3.17b}$$

Proof. The structure of the proof follows the classical approach in [34] and draws from viscosity solution theory. In every case, we start by assuming that V^+ is not a viscosity solution of the Hamilton-Jacobi-Isaacs equation (3.17) and derive a contradiction of the principle of optimality stated in Lemma 3.2.

Recall from Chapter 2 that a continuous function is a viscosity solution of a Hamilton-Jacobi partial differential equation (or a variational inequality) if it satisfies the appropriate boundary condition and is both a *subsolution* and a *supersolution*. From the definition of V^+ in (3.7), considering the terminal case $t = T$, it is straightforward to check that it satisfies the boundary condition (3.17b). The subsolution and supersolution conditions require a more involved analysis.

We will first prove that V^+ is a viscosity subsolution of (3.17). Let $\psi \in C^1(\mathbb{R}^n \times (0,T))$ such that $V^+ - \psi$ attains a local maximum at (x_0, t_0); without loss of generality, assume that this maximum is 0 (that is, ψ is "touching" V^+ from above). Analogous to Definition 2.1, we say that V^+ is a subsolution of (3.17a) if, for any such ψ,

$$\max\Big\{ \min\Big\{\partial_t\psi(x_0,t_0) + H^+\left(x_0, \nabla_x\psi, t_0\right),\ l(x_0,t_0) - \psi(x_0,t_0)\Big\},\ g(x_0,t_0) - \psi(x_0,t_0)\Big\} \geq 0. \tag{3.18}$$

From the condition of local maximum and the continuity of trajectories, we know

$$V^+\big(\mathbf{x}^{\mathbf{u},\mathbf{d}}_{x_0,t_0}(t_0+\delta), t_0+\delta\big) \leq \psi\big(\mathbf{x}^{\mathbf{u},\mathbf{d}}_{x_0,t_0}(t_0+\delta), t_0+\delta\big),$$

for sufficiently small $\delta > 0$ and all $\mathbf{u} \in \mathbb{U}_{t_0}, \mathbf{d} \in \mathbb{D}_{t_0}$. For the sake of contradiction, suppose (3.18) is false. Then it must be that

$$g(x_0,t_0) = \psi(x_0,t_0) - \theta_1, \tag{3.19}$$

and, in addition, at least one of the following holds:

$$l(x_0,t_0) = \psi(x_0,t_0) - \theta_2, \tag{3.20a}$$
$$\partial_t\psi(x_0,t_0) + H^+\left(x_0, \nabla_x\psi(x_0,t_0)(x_0,t_0), t_0\right) = -\theta_3, \tag{3.20b}$$

for some $\theta_1, \theta_2, \theta_3 > 0$. If (3.19) and (3.20a) are true, then by continuity of g, l and system trajectories, there exists a sufficiently small $\delta > 0$ such that for all inputs $\mathbf{u}$,$\mathbf{d}$ and for all $t_0 \leq \tau \leq t_0 + \delta$,

$$g\big(\mathbf{x}^{\mathbf{u},\mathbf{d}}_{x_0,t_0}(\tau), \tau\big) \leq \psi(x_0,t_0) - \frac{\theta_1}{2} = V^+(x_0,t_0) - \frac{\theta_1}{2}\ , \tag{3.21a}$$
$$l\big(\mathbf{x}^{\mathbf{u},\mathbf{d}}_{x_0,t_0}(\tau), \tau\big) \leq \psi(x_0,t_0) - \frac{\theta_2}{2} = V^+(x_0,t_0) - \frac{\theta_2}{2}\ . \tag{3.21b}$$

Then, incorporating this into the dynamic programming principle (3.11) we have

$$\begin{aligned} V^+(x_0,t_0) &\leq \sup_{\boldsymbol{\delta}\in\mathfrak{D}_t}\inf_{\mathbf{u}\in\mathbb{U}_t^T}\Big\{\min_{\tau\in[t_0,t+\delta]}\max\Big[l\big(\mathbf{x}_{x_0,t_0}^{\mathbf{u},\boldsymbol{\delta}[\mathbf{u}]}(\tau),\tau\big),\max_{s\in[t_0,\tau]}g\big(\mathbf{x}_{x_0,t_0}^{\mathbf{u},\boldsymbol{\delta}[\mathbf{u}]}(s),s\big)\Big]\Big\} \\ &\leq V^+(x_0,t_0)-\min\Big\{\frac{\theta_1}{2},\frac{\theta_2}{2}\Big\}, \end{aligned}$$

which is a contradiction, since $\theta_1,\theta_2>0$.

Similarly, if (3.19) and (3.20b) are true then, there exists a small enough $\delta>0$ such that (3.21a) holds and additionally, from Lemma 3.3, for all non-anticipative strategies $\boldsymbol{\delta}\in\mathfrak{D}_{t_0}$ and some input $\mathbf{u}\in\mathbb{U}_{t_0}$,

$$\psi\big(\mathbf{x}_{x_0,t_0}^{\mathbf{u},\boldsymbol{\delta}[\mathbf{u}]}(t_0+\delta),t_0+\delta\big)-\psi(x_0,t_0)\leq-\frac{\theta_3}{2}\delta.$$

Recalling that $V^+-\psi$ has a local maximum at (x_0,t_0), we obtain

$$V^+\big(\mathbf{x}_{x_0,t_0}^{\mathbf{u},\boldsymbol{\delta}[\mathbf{u}]}(t_0+\delta),t_0+\delta\big)\leq V^+(x_0,t_0)-\frac{\theta_3}{2}\delta\ .$$

Inspecting (3.11) in this case, we obtain

$$\begin{aligned} V^+(x_0,t_0) &\leq \sup_{\boldsymbol{\delta}\in\mathfrak{D}_t}\inf_{\mathbf{u}\in\mathbb{U}_t^T}\Big\{\max\Big[V^+\big(\mathbf{x}_{x_0,t_0}^{\mathbf{u},\boldsymbol{\delta}[\mathbf{u}]}(t_0+\delta),t_0+\delta\big),\max_{\tau\in[t_0,t_0+\delta]}g\big(\mathbf{x}_{x_0,t_0}^{\mathbf{u},\boldsymbol{\delta}[\mathbf{u}]}(\tau),\tau\big)\Big]\Big\} \\ &\leq V^+(x_0,t_0)-\min\Big\{\frac{\theta_1}{2},\frac{\theta_3}{2}\delta\Big\}, \end{aligned}$$

which again is a contradiction, since $\theta_1,\theta_3,\delta>0$. Therefore, we conclude that (3.18) must be true and hence V^+ is indeed a subsolution of (3.17).

We now proceed to show that V^+ is also a viscosity supersolution of (3.17), that is, for all $\psi\in C^1(\mathbb{R}^n\times(0,T))$ such that $V^+-\psi$ attains a local minimum at (x_0,t_0) (again, we can assume for convenience that this minimum is 0), it holds that

$$\max\Big\{\min\Big\{\partial_t\psi(x_0,t_0)+H^+\left(x_0,\nabla_x\psi,t_0\right),\ l(x_0,t_0)-\psi(x_0,t_0)\Big\},\ g(x_0,t_0)-\psi(x_0,t_0)\Big\}\leq 0. \quad (3.22)$$

If we suppose that (3.22) is false, then either it holds that

$$g(x_0,t_0)=\psi(x_0,t_0)+\theta_1, \quad (3.23)$$

or both of the following are true:

$$l(x_0,t_0)=\psi(x_0,t_0)+\theta_2, \quad (3.24a)$$
$$\partial_t\psi(x_0,t_0)+H^+\left(x_0,\nabla_x\psi(x_0,t_0),t_0\right)=\theta_3, \quad (3.24b)$$

for some $\theta_1,\theta_2,\theta_3>0$.

If (3.23) holds, then there is a small enough $\delta > 0$ such that for all trajectories starting at (x_0, t_0) and all $t_0 \leq \tau \leq t_0 + \delta$

$$g\big(\mathbf{x}^{\mathbf{u},\mathbf{d}}_{x_0,t_0}(\tau), \tau\big) \geq \psi(x_0, t_0) + \frac{\theta_1}{2} = V^+(x_0, t_0) + \frac{\theta_1}{2}.$$

Then the dynamic programming principle (3.11) yields

$$\begin{aligned} V^+(x_0, t_0) &\geq \sup_{\boldsymbol{\delta} \in \mathfrak{D}_t} \inf_{\mathbf{u} \in \mathbb{U}_t^T} \Big\{ \min \Big[\min_{\tau \in [t_0, t_0+\delta]} \max_{s \in [t_0, \tau]} g\big(\mathbf{x}^{\mathbf{u},\boldsymbol{\delta}[\mathbf{u}]}_{x_0,t_0}(s), s\big), \max_{\tau \in [t_0, t_0+\delta]} g\big(\mathbf{x}^{\mathbf{u},\boldsymbol{\delta}[\mathbf{u}]}_{x_0,t_0}(\tau), \tau\big) \Big] \Big\} \\ &\geq V^+(x_0, t_0) + \frac{\theta_1}{2}, \end{aligned}$$

which is a contradiction, as $\theta_1 > 0$.

If, on the other hand, (3.24) holds, then there is a small enough $\delta > 0$ such that

$$l\big(\mathbf{x}^{\mathbf{u},\mathbf{d}}_{x_0,t_0}(\tau), \tau\big) \leq \psi(x_0, t_0) + \frac{\theta_2}{2} = V^+(x_0, t_0) + \frac{\theta_2}{2},$$

and from Lemma 3.3 we have that, for some non-anticipative strategy $\boldsymbol{\delta} \in \mathfrak{D}_{t_0}$ and all inputs $\mathbf{u} \in \mathbb{U}_{t_0}$,

$$\begin{aligned} \frac{\theta_3}{2}\delta &\leq \psi\big(\mathbf{x}^{\mathbf{u},\boldsymbol{\delta}[\mathbf{u}]}_{x_0,t_0}(t_0 + \delta), t_0 + \delta\big) - \psi(x_0, t_0) \\ &\leq V^+\big(\mathbf{x}^{\mathbf{u},\boldsymbol{\delta}[\mathbf{u}]}_{x_0,t_0}(t_0 + \delta), t_0 + \delta\big) - V^+(x_0, t_0), \end{aligned}$$

recalling that $V^+ - \psi$ attains a local minimum at (x_0, t_0). With this, (3.11) gives

$$\begin{aligned} V^+(x_0, t_0) &\geq \sup_{\boldsymbol{\delta} \in \mathfrak{D}_t} \inf_{\mathbf{u} \in \mathbb{U}_t^T} \Big\{ \min \Big[\min_{\tau \in [t_0, t_0+\delta]} l\big(\mathbf{x}^{\mathbf{u},\boldsymbol{\delta}[\mathbf{u}]}_{x_0,t_0}(\tau), \tau\big), V^+\big(\mathbf{x}^{\mathbf{u},\boldsymbol{\delta}[\mathbf{u}]}_{x_0,t_0}(t_0 + \delta), t_0 + \delta\big) \Big] \Big\} \\ &\geq V^+(x_0, t_0) + \min\Big\{\frac{\theta_2}{2}, \frac{\theta_3}{2}\delta\Big\}, \end{aligned}$$

resulting in another contradiction, as $\theta_2, \theta_3, \delta > 0$. We therefore conclude that it must be that (3.22) holds and V^+ is a supersolution of (3.17).

Since we have shown that V^+ is both a viscosoty subsolution and a viscosity supersolution of the variational inequality, this completes the proof that V^+ is a viscosity solution of (3.17) with Hamiltonian H^+. Uniqueness follows from the classical comparison and uniqueness theorems for viscosity solutions (see Theorem 4.2 in [85]). □

Remark 3.2. *As in previous work [34, 48, 85], the continuity assumptions we made on l and g are stronger than strictly necessary (note that we did not make full use of them in the proof). Using standard results in ordinary differential equations, it can be proven that the assumption of Lipschitz continuity of l and g leads to Lipschitz continuity of V^+ (see [48], Proposition 1.1), which then satisfies* (3.17) *almost everywhere (Rademacher's theorem states that Lipschitz functions are almost-everywhere differentiable). If we only assume uniform continuity of l and g, V^+ still satisfies* (3.17) *in the viscosity sense.*

Theorem 3.1 is an important theoretical result, because it establishes that the value function defined in (3.7), which as we have seen encodes the optimal outcome of the time-varying reach-avoid game, can be obtained as the viscosity solution to a specific variational inequality. This enables us to compute the sought value function through a suite of available numerical methods developed for the computation of viscosity solutions to Hamilton-Jacobi equations [51–53].

The value function also has prescriptive value, because it also implicitly encodes optimal strategies for the game. Given $V^+(x,t)$, an optimal control for reaching $\mathbb{T}$ while remaining in $\mathbb{K}$ can be found almost everywhere (namely at points (x,t) where V^+ is differentiable), by choosing an element from the arg min set for (3.12) with $p := \nabla_x V$, that is:

$$\pi_u^*(x,t) \in \arg\min_{u\in\mathcal{U}} \max_{d\in\mathcal{D}} \nabla_x V \cdot f(x,u,d,t) \ . \tag{3.25}$$

At points (x,t) of non-differentiability we can theoretically obtain the controller through the subdifferential or superdifferential as per Definition 2.2 (depending on which set is non-empty):

$$\pi_u^*(x,t) \in \arg\min_{u\in\mathcal{U}} \max_{d\in\mathcal{D}} \sup_{(p,s)\in D^- V(x,t)} s + p \cdot f(x,u,d,t) \ , \tag{3.26a}$$

$$\pi_u^*(x,t) \in \arg\min_{u\in\mathcal{U}} \max_{d\in\mathcal{D}} \inf_{(p,s)\in D^+ V(x,t)} s + p \cdot f(x,u,d,t) \ . \tag{3.26b}$$

Analogously, although this is not usually needed, we could obtain an optimally adverse control-dependent disturbance at each state and time $\pi_d(x,u,t)$ by choosing elements from the arg max set corresponding to (3.25) and (3.26).

In practice, however, the optimal control policy is approximated numerically from the numerical approximation of the value function, obtained through computational partial differential equation solvers as we will presently see. This implies that derivatives are always replaced by a numerical approximation based on finite differences, as a result of which the distinction between (3.25) and (3.26) does not need to be made explicitly. Fortunately, this does not detract from the validity of the numerical approximation[3]: rather, the numerical derivatives can be seen as corresponding to some continuously differentiable function ψ "touching" V at (x,t). Control signals are thus constructed in a way that mirrors the constructive argument in the proof of Lemma 3.3.

The value function at each (x,t) encodes the optimal outcome under measurable control signals subject to non-anticipative disturbances. As discussed in Chapter 2, the purpose of the measurability requirement is to prevent artifacts in the mathematical machinery. In practice, all control signals $\mathbf{u} : [0,T] \to \mathcal{U}$ generated by the recursive application of the computed arg max will be measurable: this is ensured by the fact that the physical (or numerically simulated) controller will only change the value of the control input at a limited rate. Digital controllers have some minimum control cycle time $\delta > 0$, making $\mathbf{u}$ piecewise

[3] This is in fact one of the most valuable properties of viscosity solutions, central to their amenability to numerical analysis [50].

continuous and therefore measurable. Analog controllers are limited by their physical time constants (which is technically also true of the outputs of digital controllers, which must be transmitted through electronic circuits), and therefore their outputs are continuous in time.

Finally, applying this discrete-time numerical scheme does not in general result in a control signal $\mathbf{u}$ that achieves the ideal optimal value exactly, nor the numerically approximated optimal value. Nonetheless, it is possible to provide standard error bounds on the trajectories resulting from the discrete-time implementation of the optimal controller relative to the continuous-time analysis [47], as well as for the numerical schemes approximating the viscosity solution [50]. This enables a sound translation of the numerical computation of V into theoretical guarantees for the continuous-time system as well as the discrete-time controller implementation.[4]

3.3 Numerical Implementation

We present in this section a numerical method to compute the value function (3.7) for the time-varying reach-avoid problem, based on the result in Theorem 3.1. For conciseness, we drop the distinction between upper and lower values and Hamiltonians, as the method is equally applicable to either.

Let $\mathbf{i} \in I$ denote the index of the grid point in a discretized computational domain of a compact subset $\mathcal{X} \subset \mathbb{R}^n$ and let $k \in \{1, ..., K\}$ denote the index of each discrete time step in a finite interval $[0, T]$. Since our computation will proceed in backward time, we will let $T = t_0 > t_1 > ... > t_K = 0$. To numerically solve the variational inequality (3.17), we use the following procedure, based on a three-step update rule:

The method uses discretized values of the payoff function $\hat{l}(x_\mathbf{i}, t_k)$ and the discriminator function $\hat{g}(x_\mathbf{i}, t_k)$; $\hat{V}$ denotes the numerical approximation to V. The integral in the first update step (U1) is computed numerically using time derivative approximations. As an illustrative example, with a first order forward Euler scheme, we would have

$$\begin{aligned}\hat{V}(x_\mathbf{i}, t_k) = \; & \hat{V}(x_\mathbf{i}, t_{k-1}) \\ & + (t_{k-1} - t_k)\hat{H}\big(x_\mathbf{i}, D_x^+\hat{V}(x_\mathbf{i}, t_{k-1}), D_x^-\hat{V}(x_\mathbf{i}, t_{k-1})\big).\end{aligned} \tag{3.27}$$

The numerical scheme of Algorithm 3.1 is consistent with (3.17). $D_x^+\hat{V}, D_x^-\hat{V}$ represent the "right" and "left" approximations of spatial derivatives. For the numerical Hamiltonian $\hat{H}$, we use the Lax-Friedrichs approximation [51]:

$$\begin{aligned}\hat{H}(x_\mathbf{i}, D_x^+\hat{V}, D_x^-\hat{V}) = H\left(x_\mathbf{i}, \frac{D_x^-\hat{V} + D_x^+\hat{V}}{2}\right) \\ - \frac{1}{2}\alpha^\top(D_x^+\hat{V} - D_x^-\hat{V}).\end{aligned} \tag{3.28}$$

[4] We will not be focusing on this translation explicitly, but rather assume that these bounds have been taken into account when defining safety margins for the theoretical model.

Algorithm 3.1: Numerical Double-Obstacle HJI Solution

Data: $\hat{l}(x_{\mathbf{i}}, t_k), \hat{g}(x_{\mathbf{i}}, t_k)$
Result: $\hat{V}(x_{\mathbf{i}}, t_k)$

Initialization
for $\mathbf{i} \in I$ **do**
Init $\quad \hat{V}(x_{\mathbf{i}}, t_0) \leftarrow \max\{\hat{l}(x_{\mathbf{i}}, t_{\mathbf{0}}), \hat{g}(x_{\mathbf{i}}, t_{\mathbf{0}})\};$

Value propagation
for $k \leftarrow 1$ **to** K **do**
  for $\mathbf{i} \in I$ **do**
U1 $\quad \hat{V}(x_{\mathbf{i}}, t_k) \leftarrow \hat{V}(x_{\mathbf{i}}, t_{k-1}) + \int_{t_k}^{t_{k-1}} \hat{H}\big(x_{\mathbf{i}}, D_x^+ \hat{V}(x_{\mathbf{i}}, \tau), D_x^- \hat{V}(x_{\mathbf{i}}, \tau)\big) d\tau;$
U2 $\quad \hat{V}(x_{\mathbf{i}}, t_k) \leftarrow \min\left\{\hat{V}(x_{\mathbf{i}}, t_k), l(x_{\mathbf{i}}, t_k)\right\};$
U3 $\quad \hat{V}(x_{\mathbf{i}}, t_k) \leftarrow \max\left\{\hat{V}(x_{\mathbf{i}}, t_k), g(x_{\mathbf{i}}, t_k)\right\};$

The components of α are given by $\alpha_i = \max_{p \in \mathcal{I}} \left|\frac{\partial H}{\partial p_i}\right|$, where $\mathcal{I}$ is a hypercube containing all the values that p takes over the computational domain. With this choice of α for the Hamiltonian, the numerical scheme is stable [51].

In the numerical examples in Section 3.4, we use a fifth-order accurate weighted essentially non-oscillatory scheme [51, 52] for the spatial derivatives $D_x^+ \hat{V}$; for the time derivative $\nabla_t \hat{V}$, we use a third-order accurate total variation diminishing Runge-Kutta scheme [52]. These methods are implemented by means of the computational tools provided in [53]. It should be noted that lower order spatial and time derivative approximations can also yield a numerically stable (although less accurate) solution to (3.17) at lower computational expense [51].

It is important to stress the remarkable computational similarity of this new method to its time-invariant counterpart. Indeed, the only computational overhead is introduced by step (U3) in Algorithm 3.1, and the need to allow functions l, g and $\hat{H}$ to depend on time. As a result, as will be demonstrated in the following section, our method can compute the backwards reachable set for time-varying problems at essentially no additional cost compared to the time-invariant case.

Lastly, the optimal action for each player is implicitly obtained in solving the minimax to compute the Hamiltonian $\hat{H}$ in step (U1). It follows from Algorithm 3.1 that, starting inside a player's winning region (the reach-avoid set for the attacker and its complement for the defender), applying this optimal action at each state as a feedback policy yields a guaranteed winning strategy for the reach-avoid game.

3.4 Numerical Examples

To illustrate our proposed method for computing reach-avoid sets, we present two numerical examples. The first shows the computational procedure in a simple optimal control scenario with a moving target and a moving obstacle, and the obtained reach-avoid set is validated against the analytic result. The second example presents a two-player reach-avoid game with moving target and constraint sets; our method is benchmarked against the space-time state augmentation approach (as proposed in [89]), reaching the same computed set (within one grid cell of accuracy) at drastically lower computational cost (by two orders of magnitude).

3.4.1 Example 1: Reachability Problem

We begin with a comparatively simple geometric problem that we can use to validate the accuracy of the numerical results against the analytic ground truth. Consider the optimal control problem of guiding a massless vehicle that can move in any direction at a bounded speed (*simple motion* dynamics) trying to reach a moving target set while avoiding a moving obstacle. The problem is defined on a finite time interval $[0, T]$ with $T = 0.5$. The state $x(t) = (\mathrm{x}(t), \mathrm{y}(t))$ represents the vehicle's position on the plane, with dynamics

$$\dot{x} = vu(t), \quad u(t) \in \mathcal{U}, \tag{3.29}$$

where $v = 0.5$ is the maximum speed of the vehicle and $\mathcal{U}$ is the unit disc.

The target set is a square with side length 0.4 that moves downward (i.e. in the negative y direction) with velocity $v_{\mathcal{T}} = 1.5$. The center of the target set is initially located at $(0, 0.75)$ at $t = 0$, and reaches $(0, 0)$ at $t = 0.5$.

$$\mathcal{T}(t) = \{(\mathrm{x}, \mathrm{y}) : \max(|\mathrm{x}|, |\mathrm{y} - (0.75 - v_{\mathcal{T}} t)|) \leq 0.2\}. \tag{3.30}$$

We represent this moving target set using a signed distance function, $l(\mathrm{x}, \mathrm{y}, t) := s_{\mathbb{T}}(\mathrm{x}, \mathrm{y}, t)$.

The failure set is a square obstacle with side length 0.2 that also moves downward, with velocity $v_{\mathcal{F}} = 1$. The center of the obstacle is at $(0, 0)$ at $t = 0$, and $(0, -0.5)$ at $t = 0.5$.

$$\mathcal{F}(t) = \{(\mathrm{x}, \mathrm{y}) : \max(|\mathrm{x}|, |\mathrm{y} - (-v_{\mathcal{F}} t)|) < 0.1\}. \tag{3.31}$$

The constraint set $\mathcal{K}(t) = \mathcal{F}^{\mathsf{c}}(t)$ can then be encoded through the signed distance function $g(\mathrm{x}, \mathrm{y}, t) := s_{\mathbb{K}}(\mathrm{x}, \mathrm{y}, t)$.

Figure 3.1 shows the backward-time evolution of the reach-avoid set for the example problem described above. The lower boundary of the reach-avoid set for $t = 0.45$ consists of states from which the vehicle can meet the target set exactly at its final, lowermost position by constantly moving upward. This lower boundary progresses down in backward time (as the vehicle is given more time to arrive at this final position), but eventually gets "blocked" by the obstacle ($t = 0.3$), excluding any vehicle trajectories violating the constraint on their way to reaching the target. For earlier times ($t = 0.1$), the boundary is "pinched inwards"

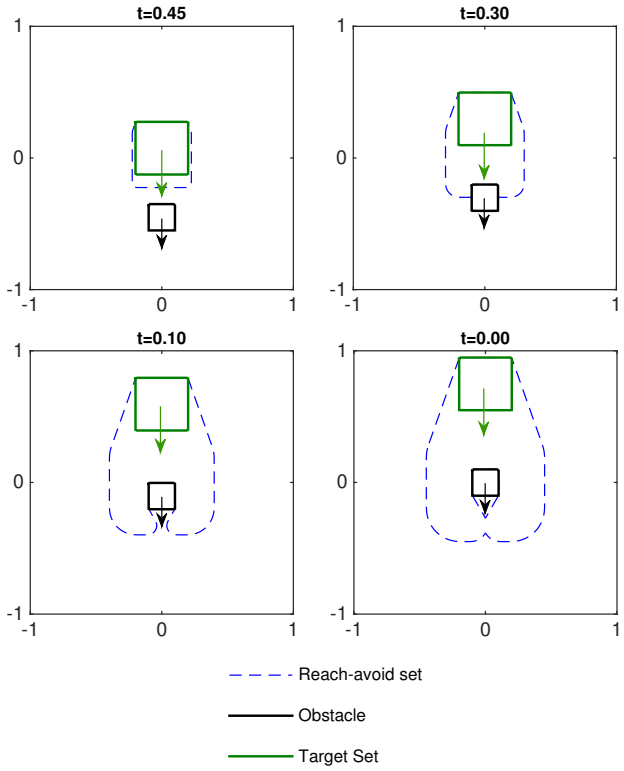


Figure 3.1: Backward-time evolution of the reach-avoid set for an optimal control problem with a target (large square) moving downward at speed 1.5, and an obstacle (small square) moving downward at speed 1. The inside of the dashed boundary represents the set of states that can reach the target set while avoiding the obstacle.

again, including nearby states from which the vehicle can move around the obstacle to safely reach the target; yet, there remains a triangular region directly below the obstacle, as seen in the $t = 0$ plot, that is not part of the reach-avoid set, because starting from those states the vehicle is unable to avoid the obstacle that is moving down. The diagonal boundaries of the reach-avoid set at its upper region are formed by those states from which the vehicle can meet the target set between its initial and final positions. Lastly, the target set is always part of the reach-avoid set, since a vehicle starting inside the target has immediately succeeded in reaching it without constraint violations (note that the target and the obstacle never overlap); conversely, the failure set is always excluded from the reach-avoid set for the opposite reason.

Analytic Solution

The reach-avoid set boundary for this example problem can be computed analytically, and thus used as ground truth for the numerically obtained boundary. Because the problem is symmetric about the y axis, we will consider the reach-avoid set in the region $x \leq 0$. We

now derive the analytic boundary by considering several different segments separately; the segments are labeled with numbers 1–7 in Figure 3.2.

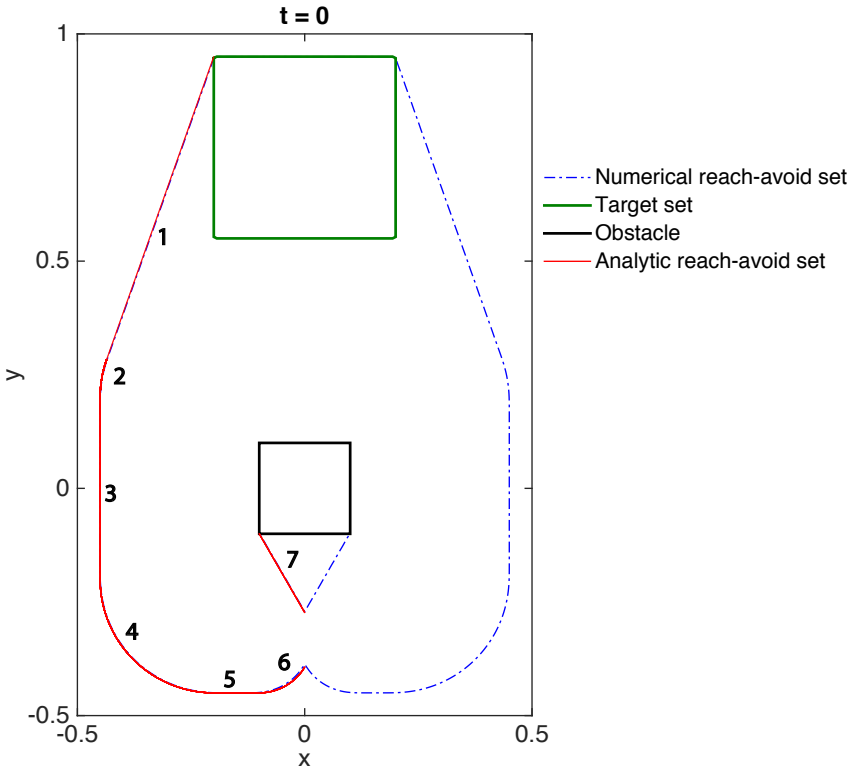


Figure 3.2: Analytic and numerical reach-avoid set. The analytic boundary is only shown on the left half of the domain to facilitate visual comparison.

Based on the uniform motion of the target and obstacle and the *simple motion* dynamics of the vehicle, we can reason about each of these segments geometrically as follows.

1. *Upper diagonal segment:* this straight segment contains states from which the vehicle can reach the top left corner of the target at an intermediate position by optimally moving in a straight line, perpendicular to the segment, towards the interception point.

$$\{(\mathrm{x},\mathrm{y}) : -0.2 + 3^{-\frac{1}{2}}(\mathrm{y} - 0.95),\ \mathrm{y} \in [0.375, 0.95]\} \tag{3.32}$$

2. *Upper transition arc:* this arc is formed by states from which the vehicle can move towards the point $(-0.1, 0.2)$ and reach the top left corner of the moving target at exactly the final time.

$$\{(\mathrm{x},\mathrm{y}) : \mathrm{x} = -0.2 - \sqrt{0.25^2 - (\mathrm{y} - 0.2)^2},\ \mathrm{y} \in [0.2, 0.375]\} \tag{3.33}$$

3. *Side vertical segment:* this straight segment contains all states from which the vehicle can reach the left side of the obstacle at its final position by traveling in a straight

horizontal line at maximum speed.

$$\{(\mathrm{x},\mathrm{y}) : \mathrm{x} = -0.45,\ \mathrm{y} \in [-0.2, 0.2]\} \tag{3.34}$$

4. *Outer lower arc:* this arc is formed by states from which the vehicle can move towards the point $(-0.1, -0.2)$ and reach the bottom left corner of the moving target at exactly the final time.

$$\{(\mathrm{x},\mathrm{y}) : \mathrm{x} = -0.2 - \sqrt{0.25^2 - (\mathrm{y} + 0.2)^2},\ \mathrm{y} \in [-0.2, -0.45]\} \tag{3.35}$$

5. *Lower horizontal segment:* this straight segment contains all states from which the vehicle can reach the lower side of the obstacle at its final position by traveling in a straight vertical line at maximum speed.

$$\{(\mathrm{x},\mathrm{y}) : \mathrm{x} \in [-0.2, -0.1], \mathrm{y} = -0.45\} \tag{3.36}$$

6. *Lower arc below obstacle:* this arc contains states from which the vehicle can move in a straight line to the point $(\mathrm{x}_c, \mathrm{y}_c) = (-0.1, -0.31)$ by time $t_c = 0.28$ (barely avoiding the obstacle's incoming lower left corner) and subsequently move vertically upwards, reaching the target at point $(-0.1, -0.2)$ at the final time.

$$\{(\mathrm{x},\mathrm{y}) : \mathrm{x} \in [-0.1, 0],\ \mathrm{y} = -0.31 - \sqrt{0.14^2 - (\mathrm{x} + 0.1)^2}\} \tag{3.37}$$

7. *Obstacle's "shadow":* this straight segment contains states from which the vehicle can barely avoid collision with the incoming obstacle by moving in a straight line to the future location of its lower left corner (analogously to Segment 1). A vehicle initially within the region enclosed by the segment and the obstacle cannot avoid a collision.

$$\{(\mathrm{x},\mathrm{y}) : \mathrm{x} \in [-0.1, 0],\ \mathrm{y} = -0.1 - \sqrt{6.25}\,(\mathrm{x} + 0.1)\} \tag{3.38}$$

Numerical Convergence

Using the scheme described in Section 3.3, we numerically solved the double-obstacle Hamilton-Jacobi variational inequality (3.17) on a computation domain consisting of $N \times N$ grid points for $N = 51, 101, 151, 201, 251, 301$. We compared each of the numerical solutions to the derived analytic solution by the following procedure:

1. Construct the (non-negative) distance function to the zero level set of the numerically computed value function encoding the boundary of the computed reach-avoid set (for instance using [53]).

2. Evaluate the distance function at approximately 20 000 points distributed on the analytically determined boundary of the reach-avoid set.

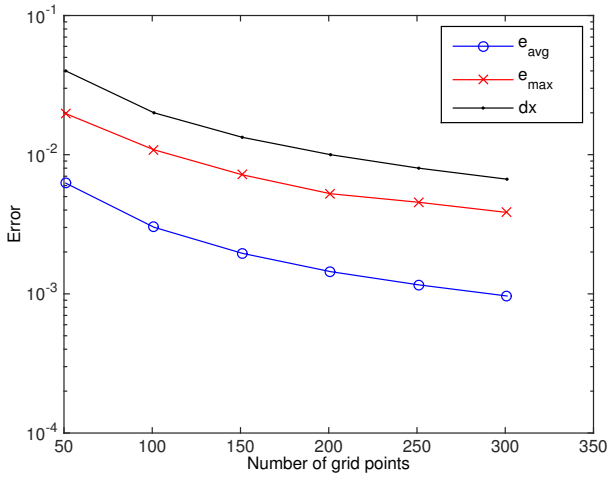


Figure 3.3: Convergence of the proposed numerical Hamilton-Jacobi solution scheme for Example 1 as the grid resolution is increased. Average error (distance) is consistently an order of magnitude smaller than the grid spacing, with the maximum error being roughly half of the grid size.

The resulting values measure the separation between points on the analytic (ground truth) reach-avoid set boundary and the numerically computed boundary. These values are used to construct error metrics for the numerical approximation.

Figure 3.3 shows in logarithmic scale the mean error and maximum error over all analytic points plotted against the number of grid points per dimension. An additional line is provided to give the scale of error in terms of the size of spatial discretization or grid spacing. Consistently across the different grid spacings, the mean error is roughly one tenth of the grid spacing, and the maximum error is approximately half of the grid spacing. The numerical scheme therefore exhibits desirable convergence both in terms of the mean error and the maximum error.

3.4.2 Example 2: Reach-Avoid Game

Figure 3.5 allows us to compare the reach-avoid sets computed by two alternative methods:

1. *Space-time state augmentation method* (4D): an augmented state representation (and numerical grid) is constructed with time $x_t \in [0, T]$ as an auxiliary state, with trivial dynamics $\dot{x}_t = 1$. All time-dependence is now reduced to state-dependence and therefore the traditional Hamilton-Jacobi-Isaacs equation [88, 93] and numerical schemes [53] for time-invariant problems can be used on this higher-dimensional system.

2. *Time-varying double-obstacle method* (3D): the problem is solved in the original state space of the problem (with a numerical grid constructed accordingly), using the formu-

lation introduced in this chapter and the numerical scheme outlined in Algorithm 3.1 for time-varying problems.

The next example introduces additional complexity with a two-player zero-sum differential game, and is used to illustrate the substantial computational savings introduced by our method relative to former approaches that required augmenting the state space with an auxiliary time variable (cf. [89]). Consider a reach-avoid game played on the finite time interval $[0, T]$, $T = 1$, during which the attacker moves freely in a square domain $[-1, 1]^2$ while the defender moves on the vertical segment $\{0.05\} \times [-1, 1]$. Let $r_A = (\mathrm{x}_A, \mathrm{y}_A)$ be the position of the attacker, and y_D be the vertical position of the defender, with the state of the system $x = (\mathrm{x}_A, \mathrm{y}_A, \mathrm{y}_D)$ governed by dynamics

$$\begin{aligned} \dot{p}_A &= v_A\, a(t), \qquad \|a\|_2 \leq 1 \ , \\ \dot{y}_D &= v_D\, b(t), \qquad b \in [-1, 1] \ . \end{aligned} \tag{3.39}$$

In this reach-avoid game, the attacker wishes to reach a target set that is moving upwards at speed $v_{\mathcal{T}} = 1.5$, while the defender tries to prevent the attacker from succeeding by intercepting or delaying its advance. The attacker is additionally required to avoid a growing obstacle whose lower edge is expanding downwards at a rate $v_{\mathcal{F}} = 0.5$. The players have maximum speeds $v_A = 2$ and $v_D = 3$. Interception is defined as the two players coming within a distance of 0.1 of each other. Figure 3.5 shows the initial configuration of the moving target and the moving obstacle, as well the interception set centered at four possible defender positions.

The reach-avoid set that we seek to compute comprises the set of joint player configurations from which the attacker is guaranteed the ability to reach the target while avoiding both interception by the defender and collision with the obstacle. In game-theoretic terms, it is the attacker's *victory domain*.

To compute the reach-avoid set, we solve (3.17) with the Hamiltonian

$$H(x, \nabla_x V, t) = \min_{\|a\|_2 \leq 1} \max_{b \in [-1,1]} \nabla_{r_A} V \cdot v_A\, a(t) + \nabla_{\mathrm{y}_D} V \cdot v_D\, b(t) \ . \tag{3.40}$$

It is straightforward to see that the decoupling between the motion of both players makes the order of optimizations irrelevant, that is, the minimax and the maximin are identical to each other. Therefore, Isaacs' condition holds and the upper and lower values coincide—we can simply speak of the value of the game. This implies that neither player needs (nor can benefit from) instantaneous information on the others' control input. The optimal Hamiltonian of the game can be directly expressed as

$$H(x, \nabla_x V, t) = -v_A \|\nabla_{r_A} V\|_2 + v_D |\nabla_{\mathrm{y}_D} V|. \tag{3.41}$$

Since the state space of the reach-avoid game is three-dimensional, we visualize two-dimensional cross-sections of the three-dimensional reach-avoid set at $t = 1$, taken at various initial defender positions.

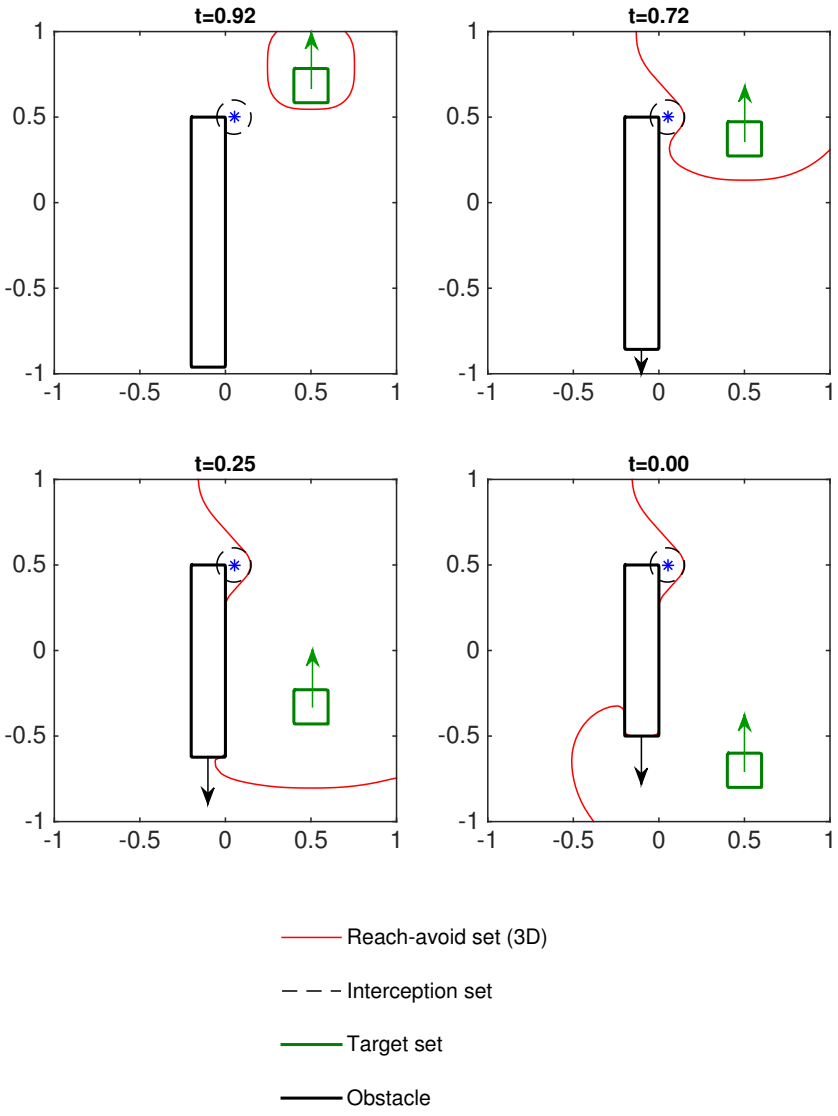


Figure 3.4: Backward-time evolution of the reach-avoid set for a reach-avoid differential game. As t decreases, the attacker has more time to reach the target, so the reach-avoid set grows. The growth of the reach-avoid set is inhibited by the defender's interception set and the obstacle.

Figure 3.4 shows multiple stages of the backward-time evolution of the reach-avoid set, sliced at the same defender position. At $t = 0.92$, there is a relatively small region in the state space from which the attacker can reach the target by the end of the game ($T = 1$). At earlier times t, the attacker has more time to reach the target and thus the reach-avoid set becomes larger; however, its growth is inhibited by the presence of the obstacle and the presence of the defender, who will actively try to intercept the attacker if it comes within range.

The shape of the reach-avoid set across the different initial defender positions is presented in Figure 3.5. If the defender starts the game near the bottom of the domain (top left plot), it will be able to block the attacker from traversing the narrowing gap below the obstacle. Thus we see that the reach-avoid set boundary does not extend into the left half quadrant of the domain. However, in this case, the attacker is free to cross the gap above the top edge of the obstacle, which leads to a large area of the top left quadrant being inside the reach-avoid set.

As the defender's starting position gradually becomes higher, we see a shift towards the opposite reach-avoid set layout with the bottom gap now becoming less well protected. However, the reach-avoid set extends into the bottom left quadrant to a lesser extent than in the top left quadrant. This is due to the fact that the passage under the obstacle is closing and the target is moving away from it: therefore an attacker not starting close enough to the opening will either get blocked out or not be able to make it through in time to reach the target.

Computational Efficiency

Computations for both methods were run in MATLAB using [53] on a computer with a Core i7-2640M processor. As can be appreciated in the figure, the reach-avoid set boundaries computed by the two methods are extremely similar (in fact, the computed reach-avoid boundaries are well within a grid cell of each other throughout the state space); this similarity contrasts with the very different amounts of computation required to obtain the two solutions. The space-time state augmentation method took approximately 1 hour and 50 minutes on a 45^4 state grid. The time-varying double-obstacle method took approximately 3 minutes on a 51^3 state grid.

Given that the dynamic programming computation is roughly linear in the number of state grid cells, it is expected that having about 30 times fewer grid cells would lead to a comparable speedup in computation, and indeed computation was roughly 36 times faster. This scaling property is important, since the grid used for this toy problem is of moderate size. As we will see in Chapter 4, many engineering problems will require much larger grids and longer time horizons (possibly with hundreds or thousands of time steps), in which case the difference in computational cost between the two methods can be of multiple orders of magnitude.

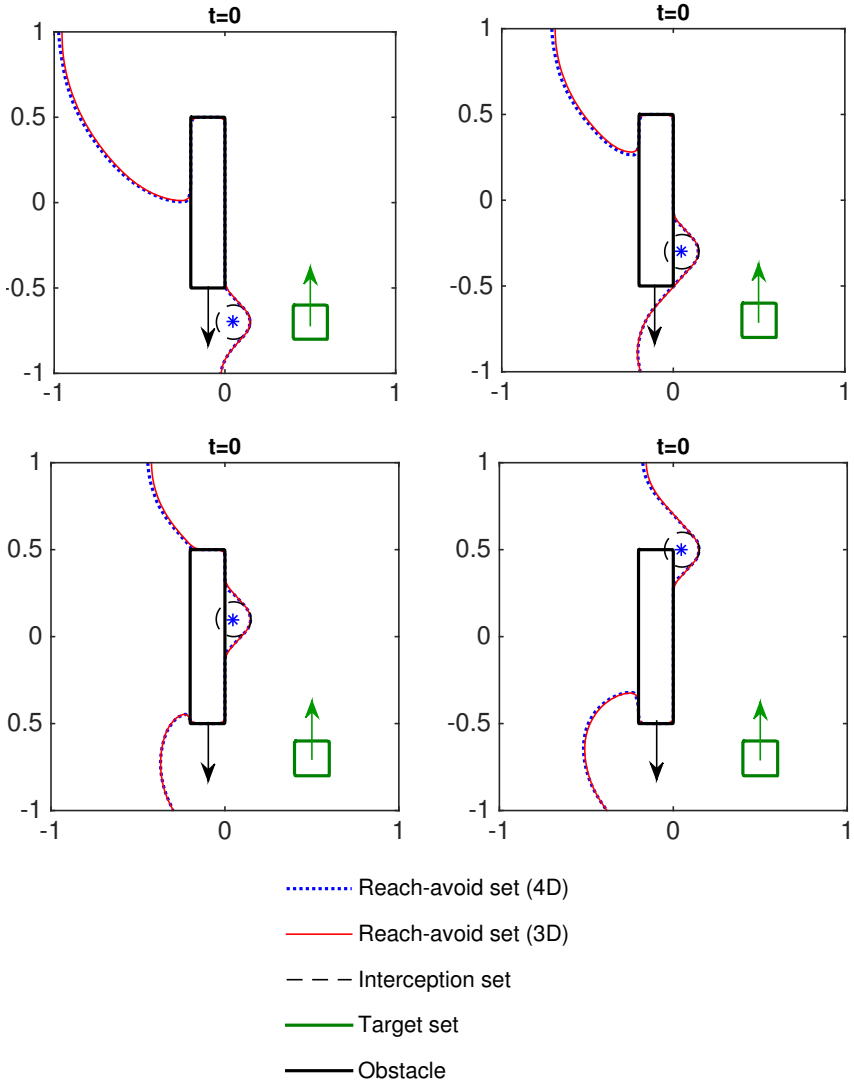


Figure 3.5: Reach-avoid set computed through the space-time state augmentation method (4D) and the time-varying double-obstacle method (3D). 2D cross-sections of the set are shown at the initial time for four different defender positions.

3.5 Chapter Summary

This chapter has introduced an extension of Hamilton-Jacobi theory and computational methods to reach-avoid problems with time-varying dynamics, targets, and constraints. This result enables the analysis of a wide range of problems in game theory and optimal control, including the study of games of pursuit and the computation of safety certificates for dynamical systems.

A critical advantage of the formulation presented here is that the numerical methods derived from it have computational complexity equivalent to that of previously existing techniques for time-invariant systems. By exploiting the structural role of time in dynamic programming, this formulation allows us to handle time variability at practically no computational cost. This sets the proposed method apart from previous approaches that work around time variation by incorporating time as an additional variable in the state.

As we will see in the Chapter 4, the time-varying Hamilton-Jacobi reach-avoid formulation directly translates into highly scalable schemes for safe planning and control in large multi-vehicle networks, by having vehicles treat certain trajectories as time-varying obstacles in their own state space.

Chapter 4

Safe Multi-Robot Trajectory Planning

> We've got a better chance of survival if we work together.
>
> Maximus Decimus Meridus
> *Gladiator*, 2000

This chapter is based on the papers "Safe Sequential Path Planning of Multi-Vehicle Systems via Double-Obstacle Hamilton-Jacobi-Isaacs Variational Inequality" [14] and "Robust Sequential Trajectory Planning Under Disturbances and Adversarial Intruder" [15], written in collaboration with Mo Chen, Somil Bansal, Shankar Sastry, and Claire Tomlin.

The problem of robot motion planning has traditionally been thought of in the context of a single robotic system operating in a static environment. This assumption has been essentially accurate for most industrial robotic applications, such as welding, painting, or palletizing, and remained mostly acceptable for low-stakes motion planning in small mobile systems like home cleaning robots [94]. However, as robotic systems begin to populate our roads and our skies, it becomes clear that they cannot be assumed to operate in isolation. Each robot's success in safely completing its task is no longer determined by its own decisions, but is also contingent on the actions of a potentially large number of other agents present in its environment.

A central component in safety assurance then becomes what the robotic system can assume about the behavior of other agents. Assuming no knowledge and attempting to ensure collision avoidance *regardless* of the behavior of all other agents is almost always a hopeless endeavor, since protecting against all their possible behaviors—and therefore against their worst-case coordinated adversarial behavior—would lead to a zero-sum game with many pursuers ganging up on a single evader. The solution to such an uneven multiplayer game (even in the rare cases in which it is computationally tractable, cf. [95]), would lead us to conclude that capture is ultimately inevitable. Of course, this is an unreasonably conservative approach under most circumstances, where most other agents do not actively seek to be

involved in a collision, and therefore opportunities for coordination can be leveraged to substantially improve safety.

In this chapter we will focus on the *cooperative* trajectory planning case, in which the multiple agents navigating the environment have the ability to share information and compute a coordinated solution. Thus, individual agents (such as autonomous mobile robots or vehicles under partial human control) may forgo part or all of their decision authority in exchange for the safety assurances obtained from coordination. For autonomous robotic systems, this coordination can be implemented through modern telecommunication technologies—including recently developed vehicle-to-vehicle (V2V) and vehicle-to-infrastructure (V2I) standards, as well as the Automatic Dependent Surveillance-Broadcast (ADS-B) system, mandatory for all commercial passenger aircraft in the United States as of January 2020 [96]. In the case of human agents, coordination must be carried out through indirect means and inevitably carries greater uncertainties for the autonomous system. Safe robotic navigation in the midst of human beings will constitute much of the focus of Part II of this thesis, in particular Chapters 7 and 8.

Even when communication is available and coordination is therefore possible in principle, the dynamic multi-agent collision avoidance problem is fundamentally challenging. We would like to find a set of trajectories that are jointly collision-free and ideally optimal or highly efficient by some performance metric, such as time, energy consumption, etc. Generally, this requires reasoning in the joint state space of all vehicles involved, which would lead computations to undergo a combinatorial explosion directly tied to the *curse of dimensionality*. Instead, we may seek to further exploit coordination to impose useful organizational structure that can enable us to tractably find efficient solutions while guaranteeing the satisfaction of safety constraints for each individual robotic system.

The sky is the limit

The last half-decade there has seen a rapid growth in the unmanned aircraft systems (UAS) industry. As UAS—or drones—continue to become more capable and economical, the number of these vehicles used for both recreational and commercial purposes continues to surge. In the United States alone, the Federal Aviation Administration (FAA) estimated a total of 1.25 million recreational drones by the end of 2018, while the number of registered commercial drones exceeded $277,000$, with an accelerating rate of new registrations per month, nearing $15,000$ in December 2018 [3]. While drones are already beginning to deliver value in a wide range of sectors, from construction and agriculture to sports and film, the proliferation of these vehicles comes with serious safety concerns. As of 2019, the FAA receives more than 100 reports every month concerning UAS being illegally operated in the vicinity of airplanes, helicopters, and airports, creating a safety liability [97]: "Reports of unmanned aircraft (UAS) sightings from pilots, citizens and law enforcement have increased dramatically over the past two years." Recent efforts to raise safety awareness among professional and recreational human operators include the Know Before You Fly educational campaign [98] and the first National Drone Safety Awareness Week scheduled for November 4–10, 2019 [99].

Both the FAA and the National Aeronautics and Space Administration (NASA) have repeatedly stressed the important challenge of integrating UAS into the national airspace. While educating human pilots and operators is a central need for the current generation of UAS, the rapidly improving autonomous decision-making capabilities on board these vehicles (with even consumer models beginning to offer fully autonomous flight functionality [100, 101]) strongly supports the search for long-term safety assurance solutions that rely heavily on automation. Since 2015, the two government agencies have been leading the UAS Traffic Management (UTM) effort [6], partnering with a number of industry players, including Google and Amazon, both of whom have been looking into autonomous UAS package delivery [102, 103]. The vision behind the UTM initiative is a scalable traffic control system that can automatically coordinate the airspace usage of hundreds or thousands of vehicles performing a wide variety of operations in the same region, ensuring that all flight trajectories are safe with respect to one another. In order for this vision to be realized, however, important technical challenges must be addressed, including the timely and reliable computation of these flight trajectories to meet the needs of large numbers of airspace users.

Related Work

The last two decades have produced a rich body of literature around the problem of safe navigation for multiple robots or vehicles. The numerous studies can roughly be categorized into two main classes: reactive approaches and planning-based approaches. Each type comes with its own particular strengths and challenges. While reactive methods tend to provide computationally cheap feedback-based rules that naturally lend themselves to real-time decision-making, it is not usually possible to derive safety guarantees from them, at least without strong assumptions on vehicle dynamics. Planning-based methods, on the other hand, are better equipped to provide formal safety guarantees, but the associated trajectory computations (often through optimization or search) scale poorly with the number of agents involved. We discuss here a few of the most relevant and representative approaches from each class.

Reactive approaches seek to achieve collision avoidance by implementing real-time decision rules for the different agents based on their current state (or observations thereof). The simplest—and earliest—reactive approaches are prescriptive protocols, where simple actions are "hard-coded" by system designers for different situations. These methods have played an important role in manned aviation for many years, with two notable examples being the Ground Proximity Warning System (GPWS), which combines direct measurements and map information to instruct pilots to pull up when flying dangerously close to terrain [104], and the Traffic Alert and Collision Avoidance System (TCAS), which monitors surrounding air traffic through active transponders (including ADS-B) and issues coordinated "resolution advisories" to pilots of different aircraft that may be at risk of mid-air collision [105]. While resolution advisories have been useful in most scenarios encountered by manned aviation, where vehicle density is limited, their use is meant for conflicts involving a small number of vehicles, and performance can degrade severely when conflicts between multiple aircraft be-

come coupled. Newer protocol-based approaches proposed for air traffic control applications have shown promise in scaling to larger vehicle numbers [106]; the associated theoretical guarantees assume that aircraft can instantaneously change speed and heading, which may be an acceptable simplification under the large vehicle separation (multiple nautical miles) that characterizes manned air traffic, but does not translate well to the close-range flight encountered in small unmanned systems, where vehicle dynamics become more relevant.

Another important family of reactive collision avoidance methods make use of the *virtual force field* approach, which modifies vehicle control actions so as to emulate a repulsive force with respect to other vehicles and obstacles [107–109]. Unfortunately these methods tend to lack dynamic feasibility guarantees, and can exhibit poor behavior especially in dense multi-agent environments: for example, the system may evolve into a state from which the forces required to avoid collisions exceed the vehicles' control authority. To address this, many modern force-field-type methods utilize the notion of *velocity obstacles* or *collision cones* [110], induced by the anticipated future positions of objects based on their current motion, applying a first-order lookahead [111–114]. These methods are no longer purely reactive, since they introduce a predictive element that can help avoid aggressive (or dynamically infeasible) last-second avoidance maneuvers by detecting potential conflicts early and encouraging vehicles to steer clear of them. However, the relatively simplistic first-order lookahead can be overly conservative and fail to find a safe course of action even when one exists (for example, if avoiding a collision with one vehicle requires temporarily entering a collision course with another).

Instead, planning approaches explicitly reason about the future motion of the different agents in the environment, and attempt to compute a joint set of trajectories such that, when executed by the corresponding vehicles, will lead to collision-free navigation. An advantage of these explicit methods is that it becomes possible to simultaneously encode other desirable properties, such as liveness (all agents completing their intended tasks without becoming deadlocked) and efficiency (energy consumption, time, etc.). Many existing methods for multi-agent motion planning come from the robotics literature, and are based on computing a geometric path via search or random sampling, implicitly assuming that the robot will be able to follow this path through simple kinematic motion [115, 116]. This assumption, typically admissible in the context of robotic manipulation, can become problematic in the case of vehicles with nontrivial, often uncertain dynamics—we will return to this point in Chapter 5, where a more detailed review of robotics motion planning methods is also provided.

The solution proposed in this chapter can be best categorized as a planning-based approach, since it reasons explicitly about the long-term evolution of agent trajectories; however, its Hamilton-Jacobi dynamic programming machinery also produces a reactive feedback control policy that ensures the desirable long-term behavior and ultimately makes safety guarantees possible even in the face of bounded disturbances and modeling error. Hamilton-Jacobi analysis has previously been used for pairwise collision avoidance between individual vehicles and platoons [23, 49], but the scalability limitations of Hamilton-Jacobi computations have thus far made it challenging to use this methodology to reason about large

numbers of vehicles.

Computational tractability is a common challenge faced by planning-based approaches when attempting to reason about the joint motion of multiple agents, since the complexity of solving a search or optimization problem is exponential in the dimensionality of the problem space, and therefore with the number of agents involved. The decomposition approach introduced in this chapter is in the spirit of the early robotics work [117], which proposed assigning priorities to multiple objects and computing their trajectories sequentially, with previous objects as space-time constraints. More recent approaches have also built on this idea, combining it with more modern motion planning techniques such as probabilistic roadmaps [118]. Rather than relying on a space-time search assuming simple kinematic motion, we leverage the time-varying Hamilton-Jacobi machinery introduced in Chapter 3 to obtain a solution for general nonlinear dynamics, which additionally presents desirable robustness and time-efficiency properties.

Contribution

The work presented in this chapter, primarily conducted under the UTM effort, eschews the combinatorial nature of the multi-agent trajectory planning problem and enables the obtention of safe motion plans whose computation can be tractably and continually carried out as new requests arrive. First, we formulate a multi-vehicle collision avoidance problem involving N autonomous vehicles, each of which seeks to reach a specified target set while avoiding environment obstacles and collision with all other vehicles. Leveraging the first-come-first-served airspace allocation structure established by the UTM paradigm, we drastically reduce problem complexity by assigning a priority to each vehicle, treating higher-priority vehicles as time-varying obstacles of (approximately) known trajectories that need to be avoided. We then utilize the double-obstacle Hamilton-Jacobi-Isaacs equation introduced in Chapter 3 to compute reach-avoid sets to plan trajectories for vehicles in order of priority. Even under modeling inaccuracies and exogenous disturbances such as wind, vehicles can commit to their flight plans by declaring a conservative nominal trajectory and then using some reserved control authority for optimal disturbance rejection at tracking time, leading to a guaranteed tracking error bound. The resulting robust Sequential Trajectory Tracking scheme enables safe trajectory computation that scales linearly, as opposed to exponentially, with the number of vehicles. We illustrate our approach on a four-vehicle scenario and include two additional large-scale demonstrations with 50 and 200 simulated vehicles, conducted by collaborators.

4.1 Safe Multiagent Trajectory Planning

Consider N vehicles that need to navigate an environment $\mathbb{R}^{n_r}$ and assign them indices $i = 1, \ldots, N$. Let vehicle dynamics be given by

$$\dot{x}_i = f_i(x_i, u_i, d_i) \tag{4.1}$$

where $x_i \in \mathbb{R}^{n_i}$ represents the state of vehicle i, $u_i \in \mathcal{U}_i$ the control of vehicle i, and $d_i \in \mathcal{D}_i$ the disturbance experienced by vehicle i. We make the usual regularity assumptions on system dynamics and input signals (specified in Chapter 2, Section 2.1) to ensure the existence and uniqueness of trajectories. For conciseness, we will omit the time range from signal set notation wherever clear from context, simply writing $\mathbb{U}_i$ and $\mathbb{D}$ as appropriate. In addition, we denote by $r_i \in \mathbb{R}^{n_r}$ the position of vehicle i in the environment, which is implicitly a function of its state, i.e. $r_i(x_i)$.

Each vehicle i has initial state x_i^0, and aims to reach its target $\mathcal{T}_i$ by some scheduled time of arrival T_i. The target represents some set of desirable states, for example the destination of vehicle i. We may additionally specify an earliest departure time $t_i^{0-} < T_i$ for each vehicle, such that its trajectory from x_i^0 cannot start before this time. On its way to $\mathcal{T}_i$, vehicle i must avoid a static set of forbidden states, jointly represented by $\mathcal{F}_i^0 \subset \mathbb{R}^{n_i}$. This set can encode physical obstacles and forbidden locations, such as tall buildings or a no-fly zone near an airport. We may also, for example, restrict the set of velocities allowed in certain regions of the environment, such as air highways: the vehicle may be allowed to travel along the highway within a certain speed range, but not transversally across it or against the specified direction of flow.

In addition to the static failure set $\mathcal{F}_i^0$, each vehicle i must also avoid entering forbidden configurations with respect to every other vehicle $j \neq i$, such as collisions or possibly other undesirable conditions (e.g. for a multirotor aircraft, flying closely above another vehicle may be forbidden, since it could cause dangerous aerodynamic interference). We define the *danger zone* between vehicles i and j as the set $\mathcal{Z}_{ij} \subset \mathbb{R}^{n_i} \times \mathbb{R}^{n_j}$ of joint states that must be avoided. We will commonly specify the joint danger zone between any two vehicles i and j to be

$$\mathcal{Z}_{ij} = \{(x_i, x_j) : \|r_i - r_j\|_2 \leq R_c\} \ , \tag{4.2}$$

that is, no two vehicles should ever come within a distance R_c of each other. This radius can be defined to include the effective sizes of the vehicles when these are non-negligible, since $\|r_i - r_j\|_2$ only accounts for the distance between their respective frames of reference. Later on, we will revisit this definition and discuss the use of more general danger zones $\mathcal{Z}_{ij}$ and the associated computational implications. Finally, vehicles only need to avoid each other while in transit: vehicles that have not yet departed or have already arrived at their destinations are, for formal purposes, not present in the environment (in the UAS context, these vehicles have not yet entered, or have already exited, the airspace).

Given the set of N vehicles, their targets $\mathcal{T}_i$, static failure sets $\mathcal{F}_i^0$, and forbidden joint configurations $\mathcal{Z}_{ij}$, we would like to compute a controller for each vehicle i which guarantees that it will reach its target $\mathcal{T}_i$ at or before the scheduled time of arrival T_i while preserving safety (i.e. avoiding violations of $\mathcal{F}_i^0$ or $\mathcal{Z}_{ij}$).

In addition, we are interested in obtaining the latest time of departure t_i^{0+} such that vehicle i can still arrive at $\mathcal{T}_i$ on time. Suppose for a moment that vehicle i has no specified *earliest* time of departure t_i^{0-}. Then, as long as it is feasible for the vehicle to reach its target in the *absence* of all other vehicles, finding a safe and timely trajectory is always feasible

by having it depart early enough. Indeed, if the environment is expected to be crowded, and given that the total number of vehicles N is finite, vehicle i can simply depart at an arbitrarily early time (and potentially arrive very early) to "beat the traffic". In practice, however, there is always an earliest time the vehicle can possibly leave (if no other restrictions apply, the time at which the trajectory is computed); if the latest viable time of departure t_i^{0+} is earlier than the earliest possible time of departure t_i^{0-}, then arriving on time is infeasible. Commuters in the San Francisco Bay Area and other metropolitan areas will no doubt be acquainted with this fundamental limitation.

4.1.1 Sequential Decomposition through Vehicle Priorities

In general, the above optimal control problem must be solved in the joint space of all N vehicles. However, due to the high joint dimensionality, a direct dynamic programming-based solution is intractable. Instead, we propose to assign a priority to each vehicle and plan vehicle trajectories in sequence given the assigned priorities: we refer to this scheme as Sequential Trajectory Planning (STP). Without loss of generality, let vehicle j have a higher priority than vehicle i if and only if $j < i$. Under the STP scheme, higher-priority vehicles can ignore the presence of lower-priority vehicles, and perform trajectory planning without taking into account the corresponding joint danger zones. A lower-priority vehicle i, on the other hand, must ensure that it does not enter a forbidden joint configuration with any of the higher-priority vehicles $j < i$; given some (possibly uncertain) representation of its future trajectory, each higher-priority vehicle j induces a time-varying obstacle set $\mathcal{F}_i^j(t)$, which represents the possible states of vehicle i such that a collision between vehicle i and vehicle j could occur. The time-varying reach-avoid Hamilton-Jacobi formulation introduced in Chapter 3 is therefore at the heart of Sequential Trajectory Planning.

It is straightforward to see that if each vehicle i is able to plan a timely trajectory that takes it to $\mathcal{T}_i$ while avoiding the static failure set $\mathcal{F}_i^0$ and the danger zones of *higher-priority vehicles* $j < i$, then this constitutes a solution to the problem of all vehicles $i = 1, \ldots, N$ reaching their targets safely and in a timely manner. With the STP scheme, the additional structure provided by the vehicle priorities allows us to sidestep the combinatorial complexity of the joint trajectory planning problem. As we will see, each vehicle in the STP scheme plans a trajectory in sequence, reasoning solely in its own state space and without the need to explicitly consider joint configurations.

Thus, STP provides a computational approach whose complexity scales linearly with the number of vehicles in the presence of disturbances, as opposed to exponentially with a direct application of dynamic programming approaches. Further, the resulting solution is, by construction, guaranteed to preserve safety subject to the bounds $\mathcal{D}_i$ on modeling error, as well as (robustly) optimal subject to the priority ordering. While the introduction of a priority ordering may in itself introduce some suboptimality relative to the unstructured solution, it is worth stressing two important points. First, no currently known methods can compute the globally (robustly) optimal solution for more than 3-4 vehicles, let alone tens or hundreds of them. And second, in many of the practical applications that we are interested in,

the priority structure is already present as a result of other technical constraints: for example, the UAS Traffic Management (UTM) paradigm led by NASA and the FAA (which this work was framed within) implements a first-come-first-served priority criterion for handling flight plan requests [6].

In the remainder of this chapter, we present the details of the STP scheme, starting with a simplified implementation that ignores modeling error in the dynamics and then presenting a robust version through optimal disturbance rejection. The robust solution involves the use of Hamilton-Jacobi safety analysis to precompute a guaranteed tracking error bound, which will be explored in more depth in Chapter 5.

4.2 Sequential Trajectory Planning Without Disturbances

In this section, we introduce the basic STP scheme assuming that there is no disturbance affecting the vehicles, and that each vehicle plans its trajectory with exact knowledge of the future trajectories of higher-priority vehicles. Although in practice, such assumptions do not hold, the description of the core STP scheme will help introduce the key concepts and mathematical machinery used by robust STP. We also show simulation results for the basic STP scheme.

Recall that vehicle j has a higher priority than vehicle i if and only if $j < i$. In the absence of disturbances, we can write the dynamics of the vehicles as

$$\dot{x}_i = f_i(x_i, u_i) \ . \tag{4.3}$$

In STP, each vehicle i plans the trajectory to its target set $\mathcal{T}_i$ while avoiding static failure set $\mathcal{F}_i^0$ and the additional failure sets $\mathcal{F}_i^j(t)$ induced by higher-priority vehicles $j < i$. Trajectory planning is done sequentially starting from the first vehicle and proceeding in descending order of priority (vehicle 1, then vehicle 2, and so on through vehicle N) so that each of the trajectory planning problems can be done in the state space of only one vehicle. During its trajectory planning process, vehicle i ignores the presence of lower-priority vehicles $k > i$, and induces the obstacles $\mathcal{F}_k^i(t)$ for vehicles $k > i$.

From the perspective of vehicle i, each higher-priority vehicle $j < i$ induces a time-varying region that vehicle i needs to keep out of. In the general case, for an arbitrary definition of the danger zone $\mathcal{Z}ij$, we can define this induced failure set $\mathcal{F}_i^j(t) \subset \mathbb{R}^{n_i}$ as:

$$\mathcal{F}_i^j(t) := \left\{ x_i \in \mathbb{R}^{n_i} : \left(x_i, \mathbf{x}_j(t)\right) \in \mathcal{Z}_{ij} \right\} \ . \tag{4.4}$$

In the case of the proximity-based danger zone $\mathcal{Z}_{ij}$ defined by (4.2), the trajectory of vehicle j can be seen as inducing a time-varying obstacle in position space $\mathcal{O}(t) \in \mathbb{R}^{n_r}$ that all lower-priority vehicles must stay clear of:

$$\mathcal{O}^j(t) := \{ r \in \mathbb{R}^{n_r} : \|r - \mathbf{r}_j(t)\|_2 \leq R_c \} \ , \tag{4.5}$$

letting $\mathbf{r}_j(t) := r_j\big(\mathbf{x}_j(t)\big)$ characterize the position trajectory of vehicle j. For each vehicle $i > j$, this induced obstacle then defines a time-varying failure set in the corresponding state space, $\mathcal{F}_i^j(t) \subset \mathbb{R}^{n_i}$:

$$\mathcal{F}_i^j(t) = \{x_i \in \mathbb{R}^{n_i} : r_i \in \mathcal{O}^j(t)\} \ . \tag{4.6}$$

The theoretical formulation of STP applies to arbitrary danger zones $\mathcal{Z}_{ij}$ and therefore any induced failure sets in the form of (4.4). However, we will see that the commonly used separation criterion (cf. FAA aircraft separation standards) introduces some important implementation advantages regarding computational complexity, particularly when vehicles may have arbitrarily different state spaces. We will therefore pay special attention to the case where induced failure sets can be expressed as (4.6).

Therefore, each vehicle i must plan its trajectory to $\mathcal{T}_i$ while avoiding the union of all the induced obstacles as well as the static failure set. The total (time-varying) failure set for vehicle i, denoted $\mathcal{F}_i(t) \subset \mathbb{R}^n{}_i$, is then defined to be the union of all the failure sets that vehicle i must avoid on its way to $\mathcal{T}_i$:

$$\mathcal{F}_i(t) = \mathcal{F}_i^0 \cup \bigcup_{j=1}^{i-1} \mathcal{F}_i^j(t) \ . \tag{4.7}$$

We can similarly define the space-time failure set for vehicle i in the usual way, following (3.2).

Each higher priority vehicle $j < i$ plans its trajectory while ignoring vehicle i. Since trajectory planning proceeds sequentially in descending order or priority, the vehicles $j < i$ will have planned their trajectories before vehicle i does. Thus, in the absence of disturbances, $r_j(t)$ is *a priori* known, and therefore $\mathcal{F}_i^j(t), j < i$ are known, deterministic moving obstacles, which means that $\mathcal{F}_i(t)$ is also known and deterministic. Therefore, the trajectory planning problem for vehicle i can be solved by first computing the (time-varying) reach-avoid set $\mathcal{RA}_i(t)$ under dynamics (4.3), which, following Chapter 3, Section 3.1.1, is a set-valued map $\mathcal{RA} : [t_i^{0-}, T_i] \rightrightarrows \mathbb{R}^{n_i}$ given by

$$\mathcal{RA}_i(t) = \{x_i : \exists \mathbf{u}_i \in \mathbb{U}_i \mid \exists \tau \in [t, T_i], \mathbf{x}_{i,x_i,t}^{\mathbf{u}_i}(\tau) \in \mathcal{T}_i \ \wedge \ \forall s \in [t, \tau], \mathbf{x}_{i,x_i,t}^{\mathbf{u}_i}(s) \notin \mathcal{F}_i(s)\} \ . \tag{4.8}$$

By directly applying the methodology introduced in Chapter 3, the reach-avoid set $\mathcal{RA}_i(t)$ for vehicle i can be obtained as the zero sublevel set of the value function $V_i : \mathbb{R}^n{}_i \times [t_i^{0-}, T_i] \to \mathbb{R}$ that solves (3.17), in the viscosity sense, with $l(x_i, t) := s_{\mathcal{T}_i}(x_i)$, $g(x_i, t) := -s_{\mathbb{F}_i}(x_i, t)$, and the optimal[1] Hamiltonian

$$H_i^*(x_i, p) = \min_{u_i \in \mathcal{U}_i} p \cdot f_i(x_i, u_i) \ . \tag{4.9}$$

Given $V_i(x_i, t)$, an optimal control policy $\pi_i : \mathbb{R}^{n_i} \times [t_i^{0-}, T_i]$ for reaching $\mathcal{T}_i$ while avoiding $\mathcal{F}_i(t)$ can be computed by choosing an element from the arg min set for (4.9) with $p := \nabla_{x_i} V_i$,

[1]Since we are not considering a disturbance input here, our two-player Hamilton-Jacobi-Isaacs equation simply becomes a one-sided Hamilton-Jacobi-Bellman equation, where the upper Hamiltonian H^+ reduces to the one-sided optimal Hamiltonian H^* by dropping the disturbance maximization.

that is:

$$\pi_i(x_i, t) \in \arg\min_{u_i \in \mathcal{U}_i} \nabla_{x_i} V_i \cdot f_i(x_i, u_i) \ . \tag{4.10}$$

As discussed in Chapter 3, Remark 3.2, the value function inherits Lipschitz continuity from l_i and g_i and is therefore almost-everywhere differentiable. The numerically well-behaved nature of viscosity solutions allows us to use finite-difference approximations of the gradient without the need to give points of non-differentiability of V_i a separate treatment [50].

The latest time of departure t_i^{0+} can be obtained from the time-varying reach-avoid set $\mathcal{RA}_i$ as[2]

$$t_i^{0+} := \max\{t : x_i^0 \in \mathcal{RA}_i(t)\} \ . \tag{4.11}$$

Once a viable time of departure $t_i^0 \in [t_i^{0-}, t_i^{0+}]$ is chosen (typically we will choose to depart as late as possible, $t_i^0 := t_i^{0+}$) the vehicle's planned trajectory $\mathbf{x}_i(\cdot)$ can then be computed by numerically integrating the system dynamics (4.3) from the initial conditions (x_i^0, t_i^0), using such an optimal policy π_i. This trajectory is then used to define the time-varying failure sets $\mathcal{F}_k^i$ that lower priority vehicles $k > i$ must stay clear of. The basic STP scheme is summarized in Algorithm 4.1.

Algorithm 4.1: Basic STP scheme

Data: Initial conditions x_i^0, vehicle dynamics f_i, targets $\mathcal{T}_i$, joint danger zones $\mathcal{Z}_{ij}$, static failure sets $\mathcal{F}_i^0$, control authorities $\mathcal{U}_i$

Result: Vehicle trajectories $\mathbf{x}_i : [t_i^0, T_i] \to \mathbb{R}^{n_i}$

for $i \leftarrow 1$ **to** N **do**

F Determine the total (time-varying) failure set $\mathcal{F}_i(t)$ as per (4.7);

R Compute the reach-avoid set $\mathcal{RA}_i(t)$ in (4.8) by solving (3.17) with $l(x_i, t) := s_{\mathcal{T}_i}(x_i)$, $g(x_i, t) := -s_{\mathbb{F}_i}(x_i, t)$, and H^* as in (4.9);

T Set time of departure $t_i^0 := t_i^{0+}$ as in (4.11);

X Plan the trajectory $\mathbf{x}_i(t)$ simulating dynamics f_i with control π_i as per (4.10);

O Given $\mathbf{x}_i(t)$, compute the induced failure sets $\mathcal{F}_k^i(t)$ for each $k > i$ following (4.4);

4.2.1 Computational Complexity and Efficient Implementations

In the most general case, where vehicles have arbitrarily different state spaces $\mathbb{R}^{n_i}$ and danger zones $\mathcal{Z}_{ij}$ are arbitrarily defined, the complexity of the STP scheme is quadratic in the number of vehicles, $O(N^2)$. This is due to the fact that each vehicle i must compute, in Step **F**, the union of the static failure set $\mathcal{F}_i^0$ with the $i-1$ induced failure sets $\mathcal{F}_i^j$ for $j = 1, \ldots, i-1$, as dictated by (4.4). Once its trajectory has been determined, we must in addition compute the $N - i$ failure sets $\mathcal{F}_k^i$ it induces for lower priority vehicles $k = i+1, \ldots, N$. This leads

[2] The set-valued map $\mathcal{RA}_i$ is upper hemicontinuous by construction and therefore the supremum is attained; equivalently, we can see that the largest t with $V(x_i^0, t) \leq 0$ is always attained, by continuity of V.

to $O(N)$ computation for each of the N vehicles, and therefore $O(N^2)$ computation overall. Nonetheless, it is important to observe that Steps **F** and **O** involve comparatively inexpensive operations requiring the characterization of failure sets; the most computationally intensive operation is the obtention of the reach-avoid set $\mathcal{RA}_i(t)$ via dynamic programming in Step **R**, which is invariably performed in the state space of a single vehicle, and therefore, unlike Steps **F** and **O**, has complexity $O(1)$.

Fortunately, there are important cases in which it is not at all necessary for each vehicle to reason separately about every preceding and ensuing vehicle. Instead, we are able to compute induced failure sets recursively with a constant amount of computation per vehicle.

Let us start with the special case in which danger zones are given in terms of separation, as in (4.2). In this case, the induced failure set $\mathcal{F}_i^j$ can be readily obtained for each vehicle i from the induced obstacle $\mathcal{O}^j$. Explicitly considering the function $r_i : \mathbb{R}^{n_i} \to \mathbb{R}^{n_r}$ determining the position corresponding to every vehicle state x_i, we have from (4.6) that $\mathcal{F}_i^j$ is in fact the preimage of $\mathcal{O}^j$ under r_i:

$$\mathcal{F}_i^j = r_i^{-1}\big(\mathcal{O}^j\big) \ . \tag{4.12}$$

By the properties of preimages, this means that we can compute the union of induced failure sets directly as the preimage of the union of induced obstacles in position space. Therefore, we can rewrite (4.7) as

$$\mathcal{F}_i(t) = \mathcal{F}_i^0 \cup r_i^{-1}\left(\bigcup_{j=1}^{i-1} \mathcal{O}^j(t)\right) \ . \tag{4.13}$$

Thanks to this observation, it is possible to streamline Steps **F** and **O** as follows: Each vehicle i is given an aggregate (time-varying) induced obstacle in position space encoding all positions that incur a violation of a danger zone with the declared trajectories of some of the preceding vehicles (for the highest-priority vehicle $i = 1$, this corresponds to the empty set). Let us call this aggregate obstacle $\mathcal{O}_i$, since it will be used to determine $\mathcal{F}_i$:

$$\mathcal{F}_i(t) = \mathcal{F}_i^0 \cup r_i^{-1}\big(\mathcal{O}_i(t)\big) \ . \tag{4.14}$$

Thus Step **F** is now implemented through a constant-time operation, i.e. it does not depend on the number of vehicles preceding vehicle i on the priority queue.

Now, once vehicle i has computed its trajectory, it need only compute the induced obstacle in position space as per (4.5) and incorporate it into the aggregate obstacle set that will be passed on to vehicle $i + 1$, that is:

$$\mathcal{O}_{i+1}(t) = \mathcal{O}_i \cup \mathcal{O}^i(t) \ . \tag{4.15}$$

Therefore, the new Step **O** is also a constant-time operation, since it no longer depends on the number of vehicles following vehicle i on the priority queue.

With this efficient implementation, mediated by the recursively augmented induced obstacle $\mathcal{O}_i(t)$, the complexity of the STP scheme becomes linear in the number of vehicles, $O(N)$, with each of the vehicles performing $O(1)$ computation to plan its trajectory.

4.2.2 Efficient Implementation for Alternative Danger Zone Definitions

The efficient implementation of computations can be extended to a broader class of scenarios, relaxing the assumption of the simple separation-based danger zones defined in (4.2). For example, the Federal Aviation Administration and International Civil Aviation Organization tend to define cylindrical separation rules, with a horizontal radius and a vertical range. It is straightforward to see that replacing the ball of radius R_c by an arbitrary set in relative position space preserves the ability to encode $\mathcal{F}_i^j$ through a position obstacle $\mathcal{O}^j$, and therefore (4.12)–(4.15) apply directly.

Further, we can continue to use the notion of position obstacle even in cases where the definition of the danger zones depends on the specific shape, size, and non-position states of each vehicle, provided that these danger zones can ultimately be encoded through some form of spacial occupancy. For example, we may have large vehicles requiring greater separation, or for which angular orientation is relevant (say, an elongated fixed-wing aircraft, which occupies a different amount of space in each direction).

We may then define a set-valued *footprint* map $\mathcal{H}_i : \mathbb{R}^{n_i} \rightrightarrows \mathbb{R}^{n_r}$ that, rather than assigning a single position r_i to each state x_i of the vehicle, assigns it a set of occupied positions. The definition of the danger zone in (4.2) can then be readily generalized to

$$\mathcal{Z}_{ij} = \{(x_i, x_j) : \min_{r_i \in \mathcal{H}_i} \min_{r_j \in \mathcal{H}_j} \|r_i - r_j\|_2 \leq R_c\} \ , \tag{4.16}$$

where, as before, the dependence of $\mathcal{H}_i$ on x_i is kept implicit for conciseness.

We can then define the footprint-based obstacle of a given vehicle j using the signed distance function to $\mathcal{H}_i(t)$

$$\mathcal{O}^j(t) := \{r \in \mathbb{R}^{n_r} : s_{\mathcal{H}_i(t)}(r) \leq R_c\} \ . \tag{4.17}$$

This footprint-based obstacle $\mathcal{O}^j$ captures the set of positions that, if occupied by any vehicle i, would lead to a violation of the danger zone $\mathcal{Z}_{ij}$. We can then recursively keep track of the aggregate footprint-based obstacle $\mathcal{O}_i$ induced by all vehicles $j = 1, \ldots, i-1$ by a process analogous to (4.15). It only remains to determine, for each vehicle i, the conversion from the aggregate footprint-based obstacle $\mathcal{O}_i(t) \in \mathbb{R}^{n_r}$ to the failure set $\mathcal{F}_i(t) \in \mathbb{R}^{n_i}$. This can be done analogously to (4.14) as

$$\mathcal{F}_i(t) = \mathcal{F}_i^0 \cup \left\{x_i : \min_{r \in \mathcal{H}_i} \min_{r_j \in \mathcal{H}_j} \|r - r'\| \leq R_c\right\} \ . \tag{4.18}$$

With this, both Step **F** and Step **O** can be implemented in $O(1)$ for each vehicle, maintaining an overall complexity of $O(N)$ for the STP scheme under a broader and practically relevant class of danger zone specifications.

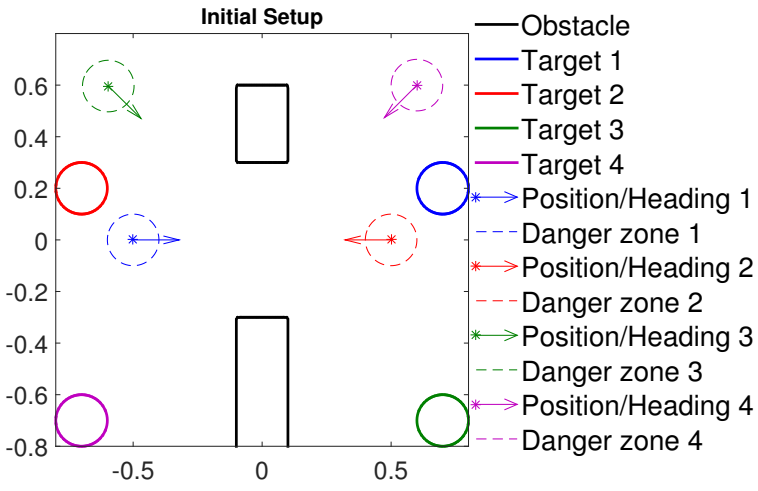


Figure 4.1: Initial configuration of the four-vehicle example.

4.2.3 Numerical Simulations

We now illustrate the basic STP scheme using a four-vehicle example. In this example, we will use the following dynamics for each vehicle:

$$\begin{aligned} \dot{\mathrm{x}}_i &= v_i \cos\psi_i \\ \dot{\mathrm{y}}_i &= v_i \sin\psi_i \qquad\qquad |\omega_i| \le \bar{\omega} \\ \dot{\psi}_i &= \omega_i \end{aligned} \tag{4.19}$$

where $x_i = (\mathrm{x}_i, \mathrm{y}_i, \psi_i)$ is the state of vehicle i, $r_i = (\mathrm{x}_i, \mathrm{y}_i)$ is the position, ψ_i is the heading, v_i is the speed, and ω_i is the turn rate. In this example, we assume that the vehicles have constant speed $v_i = 1, \forall i$, and that the control of each vehicle i is given by $u_i = \omega_i$ with $|\omega_i| \le \bar{\omega} = 1, \forall i$. We have chosen these dynamics for clarity of illustration; STP can handle more general systems of the form in which the vehicles have different control bounds and dynamics.

For this example, the target sets $\mathcal{T}_i$ of the vehicles are circles of radius r in the position space; each vehicle is trying to reach some desired set of positions. In terms of the state space x_i, the target sets are defined as $\mathcal{T}_i = \{x_i : \|r_i - c_i\|_2 \le r\}$, where c_i are centers of the target circles. For the simulation of the basic STP scheme, we used $r = 0.1$. The vehicles have target centers c_i, initial conditions x_i^0, and scheduled times of arrivals T_i as follows:

$$\begin{aligned} c_1 = (0.7, 0.2), &\quad x_1^0 = (-0.5, 0, 0), &&\quad T_1 = 0 \\ c_2 = (-0.7, 0.2), &\quad x_2^0 = (0.5, 0, \pi), &&\quad T_2 = 0.2 \\ c_3 = (0.7, -0.7), &\quad x_3^0 = (-0.6, 0.6, 7\pi/4)\,, &&\quad T_3 = 0.4 \\ c_4 = (-0.7, -0.7), &\quad x_4^0 = (0.6, 0.6, 5\pi/4)\,, &&\quad T_4 = 0.6 \end{aligned} \tag{4.20}$$

The setup for this example is shown in Figure 4.1, which also shows the static obstacles as the black rectangles around the center of the domain. The joint state space of this

four-vehicle system is twelve-dimensional (12D), making the joint trajectory planning and collision avoidance problem intractable for direct analysis using Hamilton-Jacobi reachability. Therefore, we apply the STP scheme described in Algorithm 4.1 and repeatedly solve the double-obstacle Hamilton-Jacobi equation (3.17) to obtain the optimal control for each vehicle to reach its target while avoiding higher-priority vehicles.

The simulation results can be seen in Figures 4.2, 4.3, and 4.4. Since the state space of each vehicle is 3-dimensional, each of the reach-avoid sets $\mathcal{RA}_i(t)$ is also a 3-dimensional object. For visualization purposes, we represent a 2-dimensional slice of reach-avoid sets at the initial heading angles ψ_i^0. Figure 4.2 shows the 2-dimensional backward-reachable set slices for each vehicle at its latest time of departure $t_1^{0+} = -1.12, t_2^{0+} = -0.94, t_3^{0+} = -1.48, t_4^{0+} = -1.44$, as determined by the STP scheme. The obstacles in the domain, encoded through a static failure set $\mathcal{F}_i^0 = \mathcal{F}^0 \subset \mathbb{R}^3$ common to all vehicles, and the time-varying failure sets $\mathcal{F}_i^j(t)$ induced by the trajectories of higher-priority vehicles inhibit the evolution of the backward-reachable sets, carving out "channels" that separate the backward-reachable sets into different "islands". One can see how these "channels" and "islands" form by examining the time evolution of the backward-reachable set, shown in Figure 4.3 for vehicle 3.

Finally, Figure 4.4 shows the resulting trajectories of the four vehicles. Of particular interest is the plot corresponding to time $t = -0.55$, which shows all four vehicles in close proximity without collision, i.e. with no vehicle entering another's danger zone (note, however, that danger zones themselves are allowed to overlap). This close proximity is an indication of the optimality of the basic STP scheme subject to the assigned priority ordering. Since no disturbances are present, the optimization for shortest transit time often results, as is the case here, in vehicles getting as close to others' danger zones as possible without entering them (in optimization terminology, the separation constraints are *active*).

The actual arrival times of vehicles i for $i = 1, 2, 3, 4$ are $0, 0.19, 0.34, 0.31$, respectively. It is interesting to note that for some vehicles, the actual arrival times are earlier than the scheduled times of arrivals T_i. This is because in order to arrive at the target by T_i, these vehicles must depart early enough to avoid major delays resulting from the induced obstacles of other vehicles; these delays would have led to a late arrival if vehicle i departed after t_i^{0+}.

Computations were done on a desktop computer with a Core i7 5820K processor and two GeForce GTX Titan X graphics processing units. The average computation time per vehicle is approximately 1 second using CUDA and GPU parallelization. Note that for the simulations in this and subsequent sections, almost all of the computation can be performed offline, i.e. at planning time. The only online computation needed at trajectory execution time is a look-up table query to determine the optimal control to be applied. For control-affine systems, the table can store the corresponding value function, whose (numerical) gradient is then used in the evaluation of the optimal control in (4.10), which amounts to an algebraic operation. For general dynamical systems, it may be preferable to directly store the optimal control in a look-up table, as determined during the offline numerical Hamilton-Jacobi computation, thereby eliminating the need to solve a possibly non-convex optimization problem online.

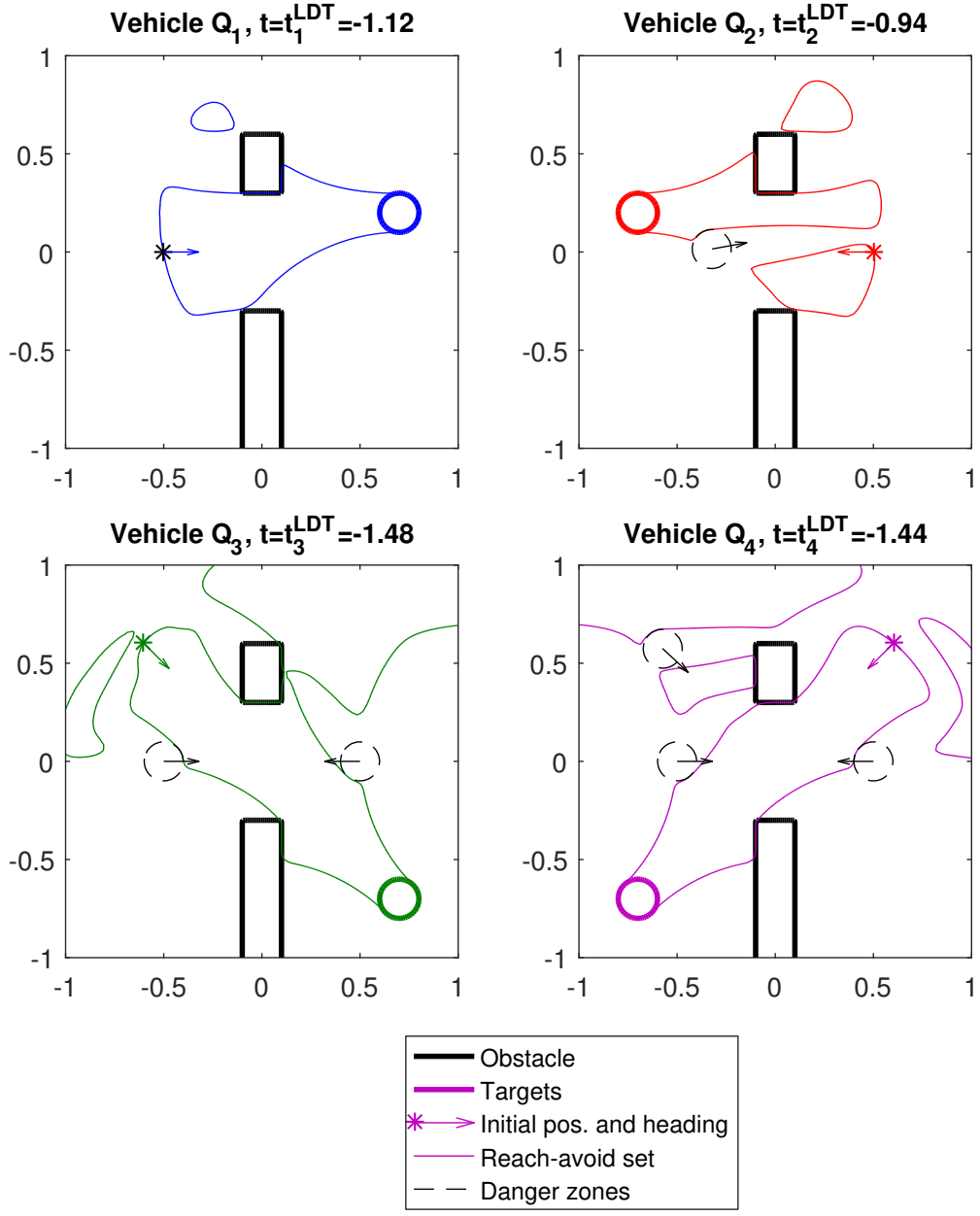


Figure 4.2: Reach-avoid sets at $t = t_i^{0+}$ for vehicles $1, 2, 3, 4$. Since each vehicle's state space is 3-dimensional, the plots present 2-dimensional slices at their respective initial headings ψ_i^0. Black arrows indicate direction of obstacle motion. Due to the turn rate constraint, the presence of static obstacles (encoded by the failure set $\mathcal{F}_i^0$) and the introduction of time-varying failure sets $\mathcal{F}_i^j(t)$ induced by higher-priority vehicles carve "channels" in the backward-reachable set, dividing it up into multiple "islands".

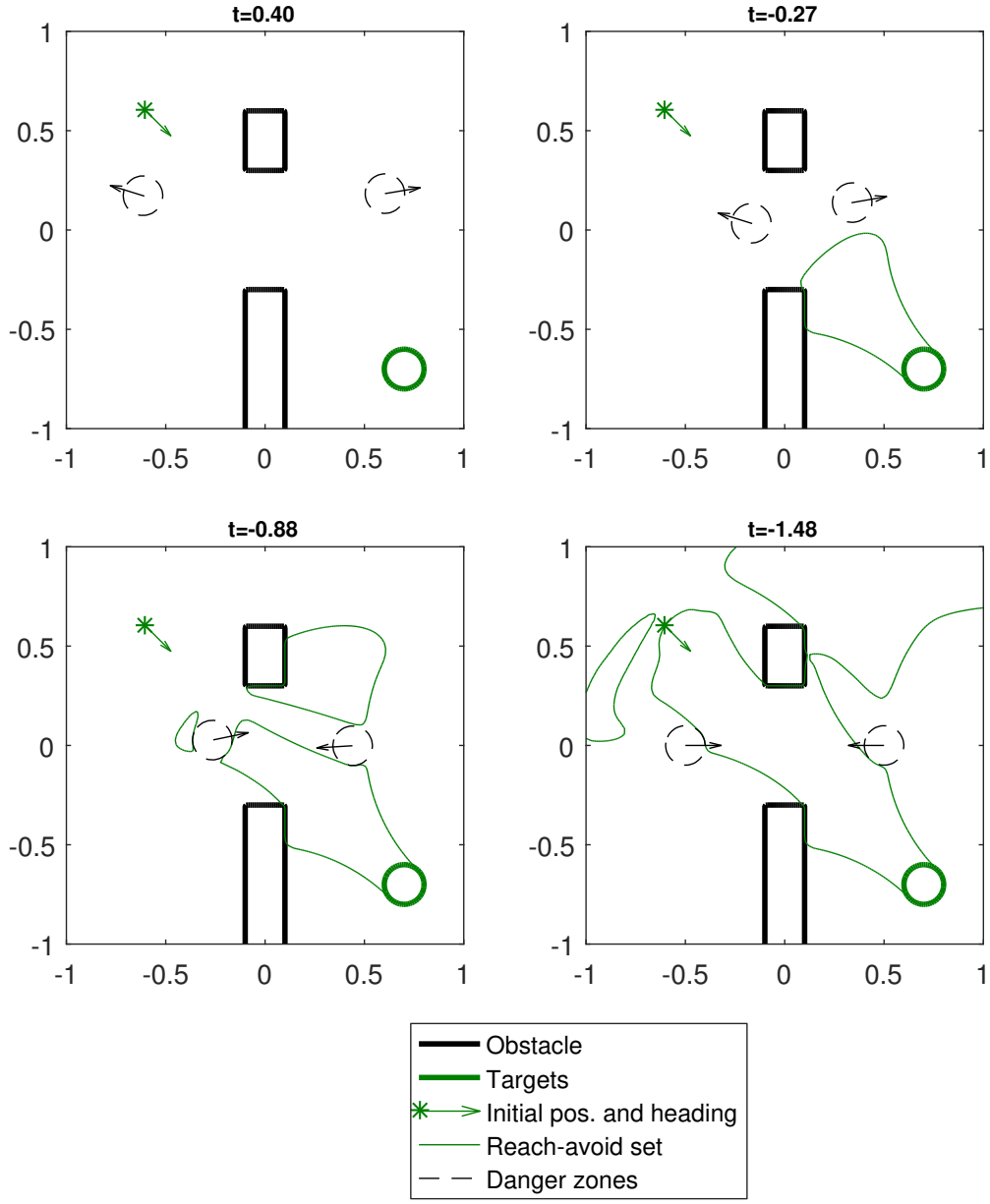


Figure 4.3: Backward-time evolution of the reach-avoid set for vehicle 3. Since the vehicle's state space is 3-dimensional, the plots present a 2-dimensional slice at its initial heading $\psi_3^0 = \frac{7\pi}{4}$. Black arrows indicate direction of obstacle motion. Top row: the reach-avoid set grows unobstructed by obstacles (time-varying and static). Bottom row: the static failure set $\mathcal{F}_i^0$ and the induced failure set $\mathcal{F}_3^1(t) \cup \mathcal{F}_3^2(t)$ carve into the backward-time propagation the reach-avoid set.

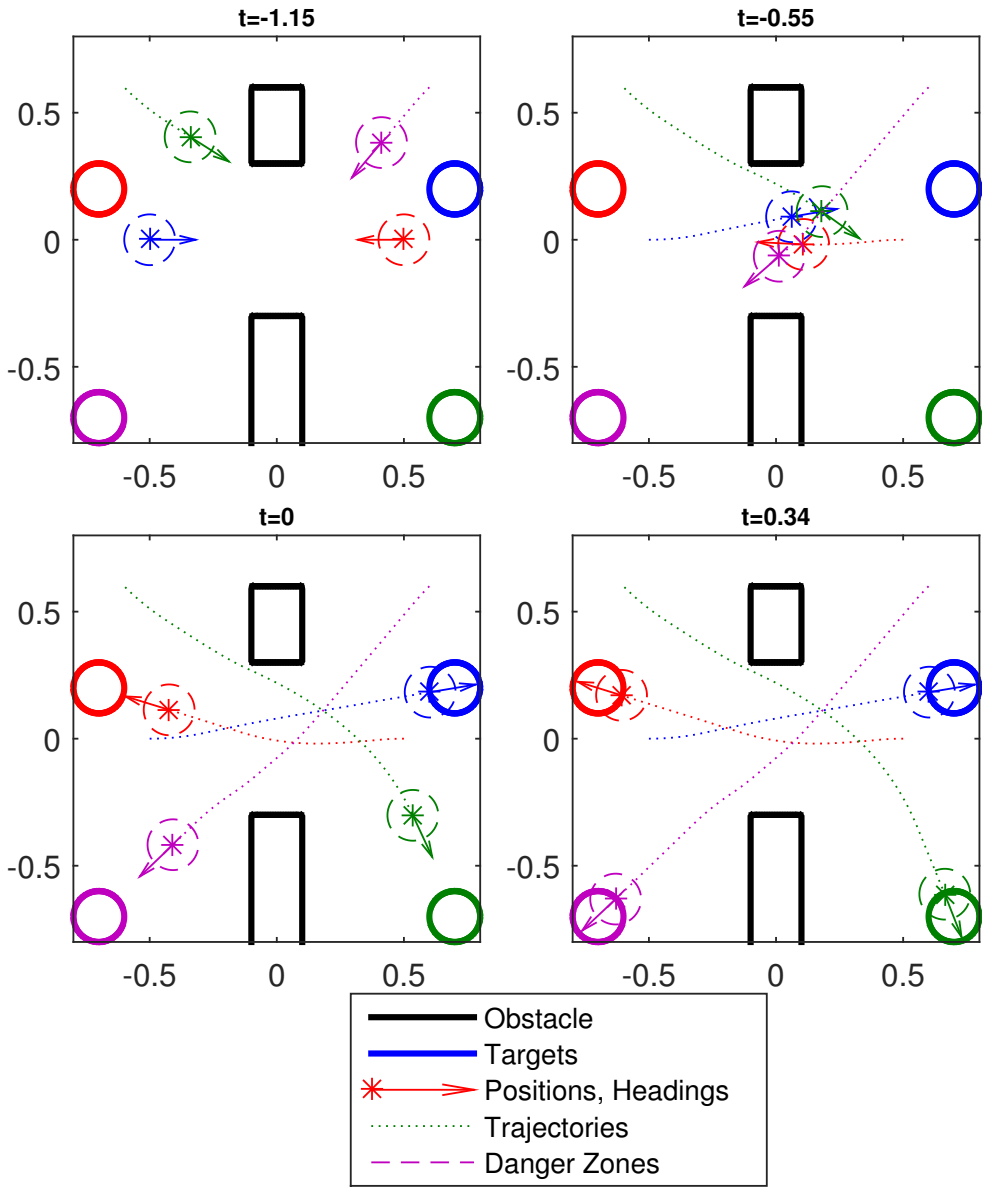


Figure 4.4: The planned trajectories of the four vehicles. Top left: only vehicles 3 (green) and 4 (purple) have started moving, showing t_i^{0+} is not common across the vehicles. Top right: all vehicles have come within very close proximity, but none is in the danger zone of another. Bottom left: vehicle 1 (blue) arrives at $\mathcal{T}_1$ at $t = 0$. Bottom right: all vehicles have reached their destination, some ahead of T_i.

4.3 Robust Tracking of Committed Trajectories

We now turn our attention to a robust STP scheme that enables us to provide assurances on the ability of real vehicles to reach their targets in a safe and timely manner, in spite of using dynamical models of limited accuracy and the vehicles potentially being subject to exogenous disturbances such as wind. This robust analysis, which can be naturally performed in the Hamilton-Jacobi-Isaacs framework, is key to the practical utility of the STP scheme.

Although it is generally impossible to perfectly track an exact trajectory under uncertain system dynamics, it may still be possible to *robustly* track a *nominal* trajectory, keeping the error between the nominal and actual state within a bounded set at all times. The resulting tracking error bound can then be used to ensure safety and success of the planned trajectory, as well as to determine the induced failure sets for lower-priority vehicles. While in this section we will assume no structural knowledge of model inaccuracies, and therefore only model the possible mismatch through a disturbance input, we will see in Section 5.1 that the central robust tracking notions introduced here extend naturally to systems with structural differences between the nominal planning model and the model used for robust tracking.

Consider the uncertain vehicle dynamics f_i as given in (4.1). The robust STP scheme carries out computation in two phases. The *trajectory planning phase* follows an analogous process to the basic STP scheme to calculate a nominal trajectory $\mathbf{x}_i^p$ in the absence of disturbances, but assuming a reduced control set $\mathcal{U}_i^p \subset \mathcal{U}_i$, effectively reserving some control authority to reject unexpected disturbances when tracking the nominal trajectory online. Prior to this phase, each vehicle independently executes the *disturbance rejection phase* to compute a worst-case bound on the tracking error under bounded disturbances $d_i \in \mathcal{D}_i$, which is then used to inform the planning. The tracking error bound is constructed so as to be enforceable by the vehicle's controller at execution time, independently from the nominal trajectory generated in the planning phase.

4.3.1 Tracking Error Dynamics

In the following, we will drop the vehicle index i when doing so does not cause ambiguity, in order to keep notation succinct. Let x and x^p denote the states taken by a vehicle and its nominal planned trajectory, respectively. Rather than directly defining the tracking error $e \in \mathbb{R}^n$ as the difference between the two, it will often be convenient to use a transformed representation

$$x - x^p = \phi(e; x^p) \ , \tag{4.21}$$

where the transformation $\phi(\cdot; x^p)$ is a bijection between e and $x - x^p$ for all x^p. As we will see momentarily, having the error representation "attached" to the frame of reference of the nominal state through ϕ can allow us to formulate error dynamics that are independent of the absolute state, which will be instrumental in precomputing a trajectory-agnostic tracking error bound ahead of time.

To this end, we also introduce transformed inputs through analogous mappings

$$u = \phi_u(u_e; x^p) \ , \qquad u^p = \phi_u^p(u_e^p; x^p) \ , \qquad d = \phi_d(d_e; x^p) \ , \tag{4.22}$$

which are also bijective for each fixed x^p. Finally, the input sets $\mathcal{U}_e, \mathcal{U}_e^p, \mathcal{D}_e$, defined as the appropriate transformations of $\mathcal{U}, \mathcal{U}^p, \mathcal{D}$, must be independent of x^p, so that the tracking error bound and tracking control policy will be valid for any absolute tracking state.

Whenever it is possible to find a transformation of this form such that the resulting error dynamics are independent of the absolute states, we will write

$$\dot{e} = f^e(e, u_e, u_e^p, d_e) \ , \tag{4.23}$$

with $u_e \in \mathcal{U}_e$, $u_e^p \in \mathcal{U}_e^p \subset \mathcal{U}_e$, and $d_e \in \mathcal{D}_e$. Importantly, the error dynamics should account for the uncertain disturbances that may affect the vehicle's trajectory $\mathbf{x}$ at tracking time, but not the ideal nominal trajectory $\mathbf{x}^p$ computed at planning time.

In fact, such a transformation exists for a broad class of vehicle dynamics, which are commonly pose-invariant in the vehicle's body frame. The following example will be useful in grounding the abstract concepts introduced so far.

Example 4.1. *For the dynamics* (4.19)*, we can represent the tracking error state by transforming the relative* (x, y) *into the nominal state's frame of reference, that is, with the planning reference always represented at the origin and facing forward.*

$$\begin{aligned} e_{\mathrm{x}} &:= (\mathrm{x} - \mathrm{x}^p)\cos\psi + (\mathrm{y} - \mathrm{y}^p)\sin\psi \\ e_{\mathrm{y}} &:= -(\mathrm{x} - \mathrm{x}^p)\sin\psi + (\mathrm{y} - \mathrm{y}^p)\cos\psi \\ e_\psi &:= \psi - \psi^p \end{aligned} \tag{4.24}$$

Under this representation, we see that the error dynamics take the form

$$\begin{aligned} \dot{e}_{\mathrm{x}} &= v\cos e_\psi + \omega^p e_{\mathrm{y}} - v^p + d_{e_{\mathrm{x}}} \\ \dot{e}_{\mathrm{y}} &= v\sin e_\psi - \omega^p e_{\mathrm{x}} + d_{e_{\mathrm{y}}} \\ \dot{e}_\psi &= \omega - \omega^p + d_\psi \end{aligned} \tag{4.25}$$

with no dependence on x or x^p. In addition, note that the input transformations ϕ_u and ϕ_u^p are simply the identity (for all x^p), whereas the disturbance transformation is

$$\begin{aligned} d_{e_{\mathrm{x}}} &:= d_{\mathrm{x}}\cos\psi + d_{\mathrm{y}}\sin\psi \\ d_{e_{\mathrm{y}}} &:= -d_{\mathrm{x}}\sin\psi + d_{\mathrm{y}}\cos\psi \\ d_{e_\psi} &:= d_\psi \end{aligned} \tag{4.26}$$

which, for a cylindrical disturbance set $\mathcal{D} := \{d : \|(d_{\mathrm{x}}, d_{\mathrm{y}})\|_2 \le \bar{d}_r, |d_\psi| \le \bar{d}_\psi\}$, satisfies the property that the transformed disturbance set $\mathcal{D}_e \equiv \mathcal{D}$ is independent of x^p.

Under this formulation of the error dynamics we would like to determine whether it is possible, and if so under what control strategy, for vehicle i to keep this error within a specified set at tracking time.

4.3.2 Bounding the Tracking Error: a Safety Problem

To obtain enforceable bounds on the tracking error, we may choose a conservative radius $R^{\mathcal{E}}$ to represent the maximum acceptable deviation between vehicle i and its nominal position throughout its trajectory. This induces an associated *tracking error bound*, defined as the constraint set

$$\mathcal{E} := \{e : \|e_r\|_2 \leq R^{\mathcal{E}}\} \ , \tag{4.27}$$

where e_r denotes the position component of e.

We can then solve a safety problem in which the controller will attempt to track the planning reference keeping the error inside the set $\mathcal{E}$ in spite of the worst-case disturbance inputs $d \in \mathcal{D}$ and, crucially, for all possible control inputs $u^p \in \mathcal{U}^p$ prescribed by the nominal trajectory. Since the tracking error bound needs to be guaranteed *before* determining a concrete nominal trajectory, it must hold for every potential trajectory plan, and we must therefore include this unknown tracking reference in our worst-case analysis of uncertainty.

Defining the *safety margin function* as the implicit surface function

$$l^{\mathcal{E}}(e) := R^{\mathcal{E}} - \|e_r\|_2 \ , \tag{4.28}$$

our safety value function, following Chapter 2, Section 2.3, is defined as

$$V^{\mathcal{E}}(e) := \inf_{\boldsymbol{\delta}_e \in \mathfrak{D}_e} \inf_{\boldsymbol{\upsilon}_e^p \in \mathfrak{U}_e^p} \sup_{\mathbf{u}_e \in \mathbb{U}_e} \inf_{t \geq 0} l_e^{\mathcal{E}}(\mathbf{e}(t)) \ , \tag{4.29}$$

with $\mathfrak{U}_e^p, \mathfrak{D}_e$ the corresponding sets of non-anticipative strategies. Using Hamilton-Jacobi-Isaacs reachability analysis, we can numerically compute the value function as the infinite-horizon limit (i.e. as $T \to \infty$ or equivalently $t \to -\infty$) of the viscosity solution $V(e,t)$ to the variational inequality

$$0 = \min\left\{\partial_t V + H^-\left(e, \nabla_e V, t\right), l^{\mathcal{E}}(e,t) - V(e,t)\right\} \ , \tag{4.30a}$$

with lower Hamiltonian

$$H^-\left(e,p,t\right) = \max_{u_e \in \mathcal{U}_e} \min_{u_e^p \in \mathcal{U}_e^p} \min_{d_e \in \mathcal{D}_e} p \cdot f^e(e, u_e, u_e^p, d_e) \tag{4.30b}$$

and terminal condition

$$V(e,T) = l^{\mathcal{E}}(e) \ . \tag{4.30c}$$

If the infinite-horizon value function converges in some neighborhood of the origin, then the zero superlevel set of $V^{\mathcal{E}}(e) := \lim_{t \to -\infty} V(e,t)$ is the maximal (infinite-horizon) controlled invariant set contained in $\mathcal{E}$, in other words, the sought *safe set* $\Omega^{\mathcal{E}}$:

$$\Omega^{\mathcal{E}} := \{e \in \mathbb{R}^n : \exists \mathbf{u}_e \in \mathbb{U}_e, \forall \boldsymbol{\upsilon}_e^p \in \mathfrak{U}_e^p, \forall \boldsymbol{\delta}_e \in \mathfrak{D}_e, \left\|\mathbf{e}^{\mathbf{u}_e, \boldsymbol{\upsilon}_e^p[\mathbf{u}_e], \boldsymbol{\delta}_e[\mathbf{u}_e]}(t)\right\| \leq R^{\mathcal{E}}, \forall t \geq 0\} \ . \tag{4.31}$$

If $\Omega^{\mathcal{E}}$ is nonempty, then the tracking error e during the flight of vehicle i is guaranteed to remain within $\Omega^{\mathcal{E}} \subseteq \mathcal{E}$ provided that the vehicle starts inside $\Omega^{\mathcal{E}}$ and subsequently applies the feedback control law

$$\pi^{\mathcal{E}}(e) = \phi_u\big(\pi_e^{\mathcal{E}}(e); x^p\big) \ , \quad \pi_e^{\mathcal{E}}(e) \in \arg\max_{u_e \in \mathcal{U}_e} \min_{u_e^p \in \mathcal{U}_e^p} \min_{d_e \in \mathcal{D}_e} \nabla_e V^{\mathcal{E}}(e) \cdot f^e(e, u, u^p, d) \ . \tag{4.32}$$

This means that the safe set $\Omega^{\mathcal{E}}$ is in itself an enforceable tracking error bound, typically one tighter than the originally postulated $\mathcal{E}$. Consequently, it will be desirable to subsequently use $\Omega^{\mathcal{E}}$ rather than $\mathcal{E}$ for planning whenever this is computationally feasible.

If the infinite-horizon value function converges but $\Omega^{\mathcal{E}}$ is empty, this will often mean that the radius $R^{\mathcal{E}}$ was not chosen conservatively enough. Fortunately, in this case there is no need to repeat the Hamilton-Jacobi computation. Since the difference $V^{\mathcal{E}} - l^{\mathcal{E}}$ is invariant to additive constants, it suffices to find a nonempty level set of $V^{\mathcal{E}}$, for some level $\alpha < 0$, and increase the value of $R^{\mathcal{E}}$ to $R^{\mathcal{E}} + |\alpha|$. With this adjustment to $R^{\mathcal{E}}$ (and thus to $l^{\mathcal{E}}$), the nonempty level set becomes the new *zero* level set of $V^{\mathcal{E}}$, leading to a nonempty $\Omega^{\mathcal{E}}$.

If, on the other hand, the infinite-horizon value function diverges to $-\infty$ everywhere, then it is not in fact possible for vehicle i to robustly track planned trajectories under these conditions, and arbitrarily large tracking errors may take place. This indicates that we did not reserve sufficient control authority for successful disturbance rejection, and must repeat the computation after appropriately reducing $\mathcal{U}^p$.

4.3.3 Trajectory Planning with the Tracking Error Bound

Once an enforceable tracking error bound has been determined, the STP scheme can proceed with some adjustments to account for this information. The *robust tracking set* of all possible states that vehicle i might find itself in when attempting to track a planning state x_i^p is

$$\tilde{\mathcal{X}}_i(x_i^p) := \{x_i^p + \phi(e_i; x_i^p),\ e_i \in \Omega_i^{\mathcal{E}}\} \ , \tag{4.33}$$

with $\phi(\cdot; x_i^p)$ the transformation map in (4.21). In cases where the tracking error state representation does not depend on x_i^p, we trivially have $\phi(e_i; x_i^p) \equiv e_i$, and therefore can simply write $\tilde{\mathcal{X}}_i(x_i^p) = x_i^p + \Omega_i^{\mathcal{E}}$, with "+" denoting the Minkowski sum.

Based on this robust tracking set, we can guarantee that vehicle i will reach its target $\mathcal{T}_i$ at flight time provided that we can plan a nominal trajectory that reaches the (reduced) robust target set

$$\tilde{\mathcal{T}}_i := \{x_i^p \in \mathbb{R}^{n_i} : \tilde{\mathcal{X}}_i(x_i^p) \subseteq \mathcal{T}_i\} \ . \tag{4.34}$$

In the special case where $\phi(e_i; x_i^p) \equiv e_i$, we can directly write $\tilde{\mathcal{T}}_i = \mathcal{T}_i - \Omega_i^{\mathcal{E}}$, with "$-$" denoting the Minkowski difference. Note that there may be cases where $\tilde{\mathcal{T}}$ is empty, implying that no nominal planning state can guarantee that the actual state of vehicle i will be in the target $\mathcal{T}_i$ at flight time (e.g. if the target set is small relative to the guaranteed tracking error).

Regarding static failure sets and time-varying failures sets induced by higher-priority vehicles, the opposite operation needs to be carried out: since the actual vehicle state x_i can

be anywhere in the set $\tilde{\mathcal{X}}(x_i^p)$, we must restrict x_i^p to stay clear of any states where any part of this set is in violation of a constraint. We therefore have the (augmented) robust failure set for vehicle i,

$$\tilde{\mathcal{F}}_i(t) := \{x_i^p \in \mathbb{R}^{n_i} : \tilde{\mathcal{X}}_i(x_i^p) \cap \mathcal{F}_i(t) \neq \emptyset\} \ , \tag{4.35}$$

which contains all the states that the planned trajectory $\mathbf{x}_i^p$ of vehicle i must not enter at time t. In the special case where $\phi(e_i; x_i^p) \equiv e_i$, this becomes $\tilde{\mathcal{F}}_i(t) = \mathcal{F}_i(t) + (-\Omega_i^{\mathcal{E}})$, where $-\Omega_i^{\mathcal{E}} := \{-x_i : x_i \in \Omega_i^{\mathcal{E}}\}$).

With these robust sets, we can define the corresponding reach-avoid set for vehicle i

$$\mathcal{RA}_i(t) = \{x_i : \exists \mathbf{u}_i \in \mathbb{U}_i^p \mid \exists \tau \in [t, T_i], \mathbf{x}_{i,x_i,t}^{\mathbf{u}_i}(\tau) \in \tilde{\mathcal{T}}_i \wedge \forall s \in [t, \tau], \mathbf{x}_{i,x_i,t}^{\mathbf{u}_i}(s) \notin \tilde{\mathcal{F}}_i(s)\} \ , \tag{4.36}$$

which is computed during the planning phase, analogously to the basic STP scheme, by solving the corresponding double-obstacle Hamilton-Jacobi-Isaacs equation (3.1), with the surface functions $l(x_i, t) := s_{\tilde{\mathcal{T}}_i}(x_i)$, $g(x_i, t) := -s_{\tilde{\mathbb{F}}_i}(x_i, t)$, and the optimal Hamiltonian

$$H_i^*(x_i, p) = \min_{u_i \in \mathcal{U}_i^p} p \cdot f_i^p(x_i, u_i) \ , \tag{4.37}$$

with $f_i^p(x_i, u_i) := f_i(x_i, u_i, 0)$ denoting the nominal dynamical model used for planning.

Finally, once a nominal planned trajectory is determined for vehicle i by simulating the dynamics f_i^p under an optimal control obtained analogously to (4.10), we wish to compute the induced failure sets $\mathcal{F}_k^i$ for all lower priority vehicles k resulting from the uncertain (but robust) tracking of vehicle i. In the general case, for arbitrary danger zones $\mathcal{Z}_{ki}$, we can extend (4.4) to

$$\tilde{\mathcal{F}}_k^i(t) := \left\{x_k \in \mathbb{R}^{n_k} : \exists x_i \in \tilde{\mathcal{X}}_i\big(\mathbf{x}_i^p(t)\big) \mid (x_k, x_i) \in \mathcal{Z}_{ki}\right\} \ . \tag{4.38}$$

With this, the robust STP scheme can proceed for all vehicles as summarized in Algorithm 4.2.

As discussed in Section 4.1, we are often interested in danger zones defined in terms of vehicle proximity, as in (4.2). Since by definition $\Omega_i^{\mathcal{E}} \subseteq \mathcal{E}_i := \{e_i : \|e_{r,i}\|_2 \leq R_i^{\mathcal{E}}\}$, we have that all states $x_i \in \tilde{\mathcal{X}}(x_i^p)$ satisfy $\|r_i - r_i^p\|_2 \leq R_i^{\mathcal{E}}$. Therefore, under this robust trajectory planning scheme, we can directly extend (4.5) as

$$\tilde{\mathcal{O}}^i(t) := \{r \in \mathbb{R}^{n_r} : \|r - \mathbf{r}_i(t)\|_2 \leq R_c + R_i^{\mathcal{E}}\} \ , \tag{4.39}$$

and, as in (4.6), the induced failure set is then obtained as

$$\tilde{\mathcal{F}}_k^i(t) = \{x_k \in \mathbb{R}^{n_k} : r_k \in \tilde{\mathcal{O}}^i(t)\} \ . \tag{4.40}$$

A similar treatment to (4.40) is also possible for defining $\tilde{\mathcal{T}}_i$ and $\tilde{\mathcal{F}}_i$ whenever targets and constraints are specified in terms of vehicle position. In general, such an approach is slightly more conservative, since we are effectively augmenting and reducing these sets by $\mathcal{E}_i$ instead of its maximal controlled invariant set $\Omega_i^{\mathcal{E}} \subseteq \mathcal{E}_i$.

We will next demonstrate the functioning of the robust STP scheme and its practical usability for dense vehicle routing in the airspace under unknown disturbances and modeling error.

Algorithm 4.2: Robust STP scheme

Data: Initial conditions x_i^0, vehicle dynamics f_i, targets $\mathcal{T}_i$, joint danger zones $\mathcal{Z}_{ij}$, static failure sets $\mathcal{F}_i^0$, control authorities $\mathcal{U}_i$, uncertainties $\mathcal{D}_i$

Result: Nominal vehicle trajectories $\mathbf{x}_i : [t_i^0, T_i] \to \mathbb{R}^{n_i}$, Robust tracking control policies $\pi_i^{\mathcal{E}} : \mathbb{R}^{n_i} \to \mathcal{U}_i$

Disturbance rejection phase

for $i \leftarrow 1$ **to** N **do**

E Set a reduced nominal control set $\mathcal{U}_i^p \subset \mathcal{U}_i$ and an error bound radius $R_i^{\mathcal{E}}$;

Ω Compute the controlled invariant set $\Omega_i^{\mathcal{E}}(t)$ in (4.31) by solving (4.30) with $l_i^{\mathcal{E}}(e_i) := R_i^{\mathcal{E}} - \|e_{r,i}\|_2$;

U Determine the robust tracking control policy $\pi_i^{\mathcal{E}}$ as per (4.32) to use online;

Trajectory planning phase

for $i \leftarrow 1$ **to** N **do**

F Determine the robust target set $\tilde{\mathcal{T}}_i$ from (4.34) and the robust total (time-varying) failure set $\tilde{\mathcal{F}}_i(t)$ as per (4.7) and (4.35);

R Compute the reach-avoid set $\mathcal{RA}_i(t)$ in (4.36) by solving (3.17) with $l(x_i, t) := s_{\tilde{\mathcal{T}}_i}(x_i)$, $g(x_i, t) := -s_{\tilde{\mathbb{F}}_i}(x_i, t)$, and H^* as in (4.37);

T Set time of departure $t_i^0 := t_i^{0+}$ as in (4.11);

X Plan nominal trajectory $\mathbf{x}_i^p(t)$ simulating f_i under π_i, as per (4.10) with $u_i \in \mathcal{U}_i^p$;

O Given $\mathbf{x}_i^p(t)$, compute robust induced failure sets $\tilde{\mathcal{F}}_k^i(t)$ for $k > i$ following (4.38);

4.3.4 Numerical Simulations

We begin with a four-vehicle example similar to the one in Section 4.2. Each vehicle has the simple kinematics model in (4.19) but with disturbances added to the evolution of each state component:

$$\begin{aligned} \dot{\mathrm{x}}_i &= v_i \cos\psi_i + d_{\mathrm{x},i} & \underline{v} \le v_i \le \bar{v} \\ \dot{\mathrm{y}}_i &= v_i \sin\psi_i + d_{\mathrm{y},i} & \|(d_{\mathrm{x},i}, d_{\mathrm{y},i})\|_2 \le d_r \\ \dot{\psi}_i &= \omega_i + d_{\psi,i}, & |\omega_i| \le \bar{\omega}, \quad |d_{\psi,i}| \le \bar{d_\psi} \end{aligned} \tag{4.41}$$

where $d = (d_{\mathrm{x},i}, d_{\mathrm{y},i}, d_{\psi,i})$ represents the disturbances affecting the three state components of vehicle i. The control of vehicle i is $u_i = (v_i, \omega_i)$, where v_i is the speed of vehicle i and ω_i is the turn rate; both controls have a lower and upper bound. For illustration purposes, we choose $\underline{v} = 0.5, \bar{v} = 1, \bar{\omega} = 1$ for all vehicles; however, the method easily handles different input bounds and even dynamical models across vehicles. The disturbance bounds are chosen as $d_r = 0.1, \bar{d_\psi} = 0.2$, which correspond to a 10% uncertainty in the dynamics.

For this example, we have chosen equal scheduled times of arrival $T_i = 0\ \forall i$. Each vehicle aims to reach a target set of radius $r = 0.1$. The vehicles' target centers c_i and initial

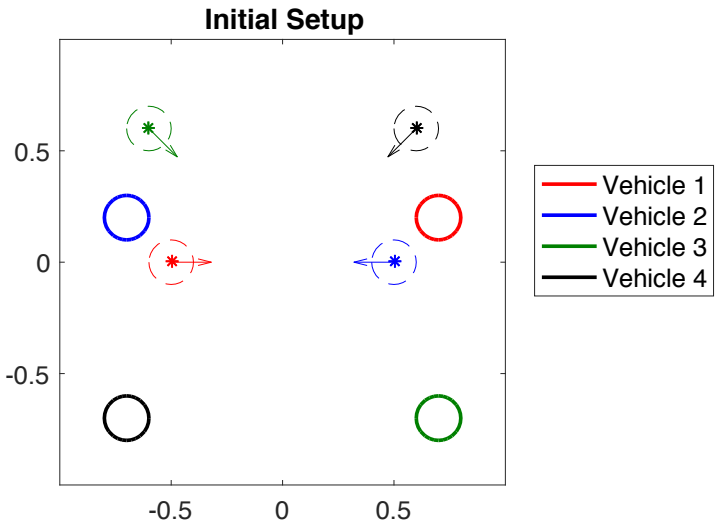


Figure 4.5: Initial configuration of the four-vehicle example in the presence of disturbances.

conditions x_i^0 are the same ones specified in (4.20). Unlike the example in Section 4.2 here we will consider no static obstacles. The problem setup for this example is shown in Figure 4.5.

After the appropriate computation, all vehicles are able to avoid each other's danger zones (colored dashed circles) and reach their target sets by their scheduled time of arrival. Offline computations were done on a desktop computer with a Core i7 5820K processor and two GeForce GTX Titan X graphics processing units. The average computation time per vehicle is approximately 2 seconds using CUDA and GPU parallelization.

Following Algorithm 4.2, we first establish a reduced control set to be used in the planning phase (which will compute trajectories assuming no disturbances): the maximum turn rate during trajectory planning is reduced from 1 to 0.6, and the speed is restricted from its default $[0.5, 1]$ range to exactly 0.75 (constant speed). The remaining control authority is not exploited in planning, and therefore saved for disturbance rejection online. Under these conditions, the disturbance rejection precomputation (Step Ω) verified that any nominal trajectory generated in the planning phase can be robustly tracked within a distance of $R^{\mathcal{E}} = 0.075$.

The planning phase is then carried out with reduced robust targets and augmented robust failure sets (as per Step **F**). Figure 4.6 shows 2-dimensional slices of the evolution of reach-avoid set and induced failure sets for vehicle 3. The smaller effective size of the robust target set $\tilde{\mathcal{T}}_3$ translates into a narrower reach-avoid set in the vicinity of the target: the planned trajectory needs to aim near the center of the original target $\mathcal{T}_3$ to ensure that vehicle 3 will successfully reach it in spite of modeling error and *a priori* unknown disturbances. Further, the obstacles induced by the nominal trajectories of vehicles 1 and 2 can be seen to be larger than in the basic STP example, and their orientation changes with the nominal bearing angle.

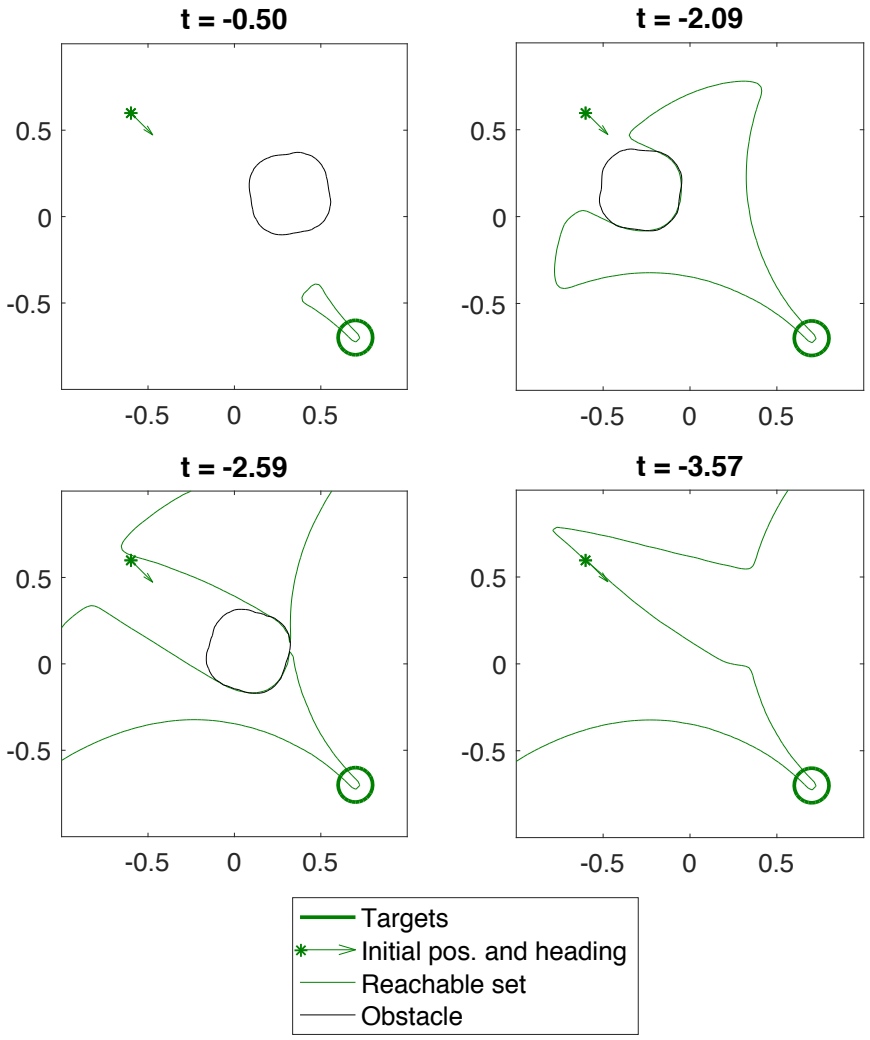


Figure 4.6: Backward-time evolution of the reach-avoid set for vehicle 3 in the robust trajectory tracking method. The larger channels carved out by the robust failure sets induced by the uncertain trajectories of higher-priority vehicles make it unsafe for vehicle 3 to depart at $t = -2.59$, and it must instead leave early ($t_3^{0+} = -3.57$) to avoid a potential collision. In addition, a smaller, robust target set is used to compute the reach-avoid set to ensure that the vehicle reaches the target set by $t = 0$ in spite of the allowed tracking error.

Figure 4.7 shows a snapshot of vehicles' trajectory execution, in addition to the entire nominal trajectories. In this case, the latest times of departure t_i^{0+} obtained for the four vehicles are $-1.61, -3.16, -3.57$ and -2.47 respectively. Vehicles leave a greater separation margin as a result of the larger robust failure sets; under certain realizations of the disturbances, they may come close to one another, but stopping short of violating the specified danger zones, as guaranteed by the Hamilton-Jacobi analysis.

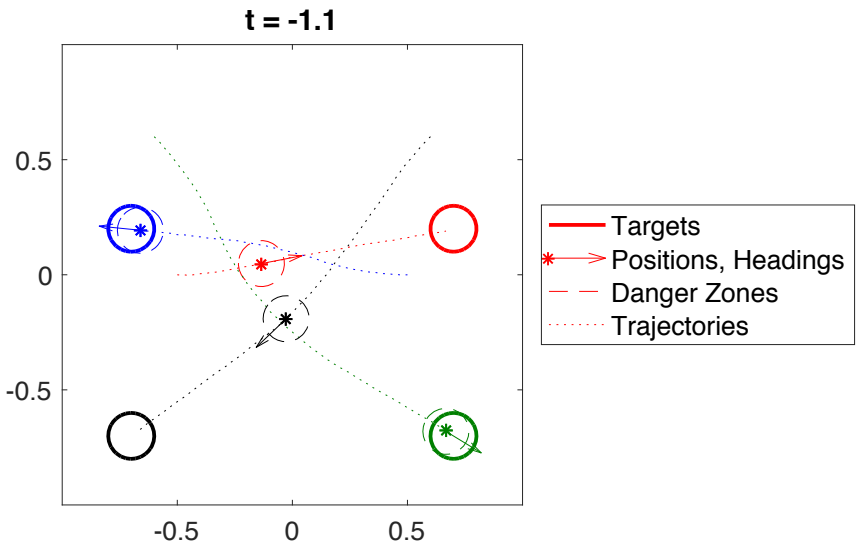


Figure 4.7: Simulated trajectories for the robust Sequential Trajectory Planning scheme in the 4-vehicle scenario.

Finally, the robust STP scheme presented here has been tested in follow-up work [119] on a larger-scale simulation considering up to $N = 200$ vehicles flying over the San Francisco Bay Area. We briefly reproduce some illustrative results here for their relevance, and point the interested reader to an in-depth evaluation in [119]. Figure 4.8 shows the trajectories planned for and tracked by 50 vehicles with dynamics (4.41) under moderate and strong breeze conditions (corresponding to forces 4 and 6 on the Beaufort wind scale). Wind disturbances affecting each vehicle were sampled uniformly at random from $\mathcal{D}_i$ at each simulation time step. The effects of maximum wind speed d_r and scheduled arrival times T_i (simultaneous vs staggered) on vehicle trajectories are demonstrated. Figure 4.9 shows the trajectories of 200 vehicles in the larger San Francisco Bay Area. Computations in these larger environments took an average of 4 minutes per vehicle using a CUDA implementation of [53] on a computer with a Core i7 5820K processor and two GeForce GTX Titan X graphics processing units.

4.4 Least-Restrictive STP: Alternative Performance Objectives

As we have seen, the Hamilton-Jacobi reach-avoid solution readily yields minimum-time trajectories by choosing to initiate trajectories at the latest possible departure time t_i^{0+}. However, as is often the case with Hamilton-Jacobi reachability solutions, the structure of the reach-avoid set and value function can be exploited to enable more flexible uses. In particular, we can use it to derive a *least-restrictive* supervisory control scheme (in the sense of [120]) that enables optimizing vehicle trajectories for an arbitrary objective of interest

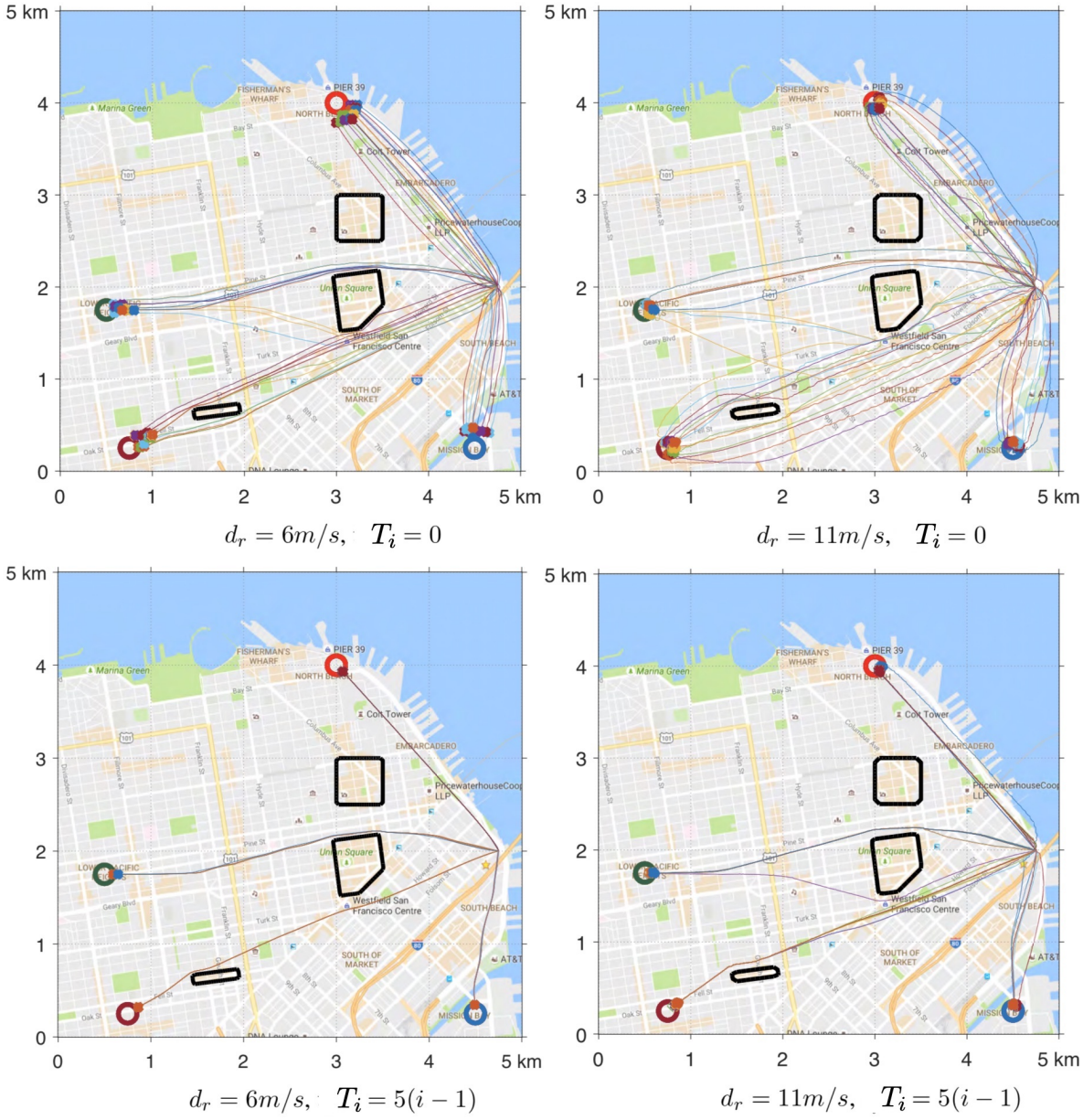


Figure 4.8: Simulated trajectories for the robust Sequential Trajectory Planning scheme in a 50-vehicle scenario over the city of San Francisco. Vehicles are assigned a common departure location and one of four different targets. Three static failure sets (no-fly zones) are indicated as black polygons on the map. The plots correspond to different wind conditions and arrival schedules: simultaneous arrival deadlines lead to the emergence of parallel "lanes" (with lower-priority vehicles departing and arriving early), whereas vehicles tend to fly along the same (time-optimal) paths when scheduled to arrive in sequence. Stronger winds increase spatial separation between vehicle trajectories. *Source:* [119] *(reproduced with permission).*

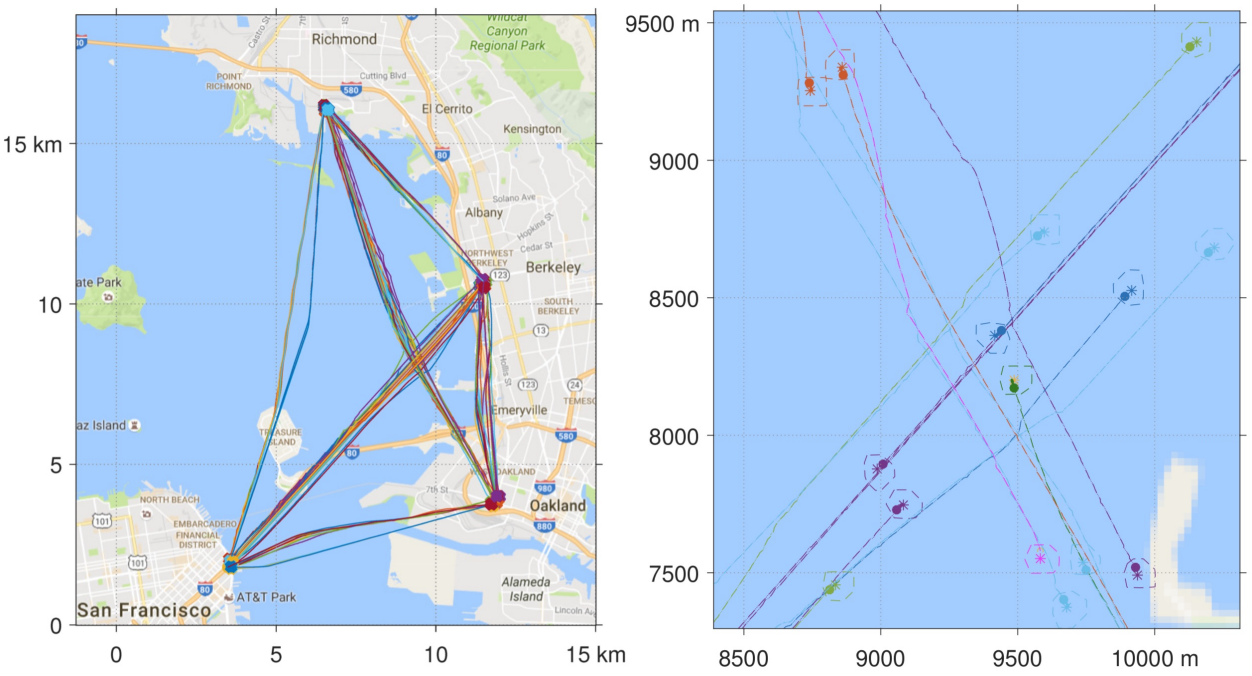


Figure 4.9: Simulated trajectories for the robust Sequential Trajectory Planning scheme in a 200-vehicle scenario over the San Francisco Bay Area. Vehicles are assigned trajectories between four different regions of interest. Vehicles are subject to independent random wind disturbances of up to $d_r = 11$ m/s and have sequential arrival schedules $T_i = 5(i-1)$. *Right:* zoomed-in snapshot. A high density of vehicles is achieved near the center, where many vehicle trajectories intersect in space. The robust STP scheme ensures separation even in spite of tracking errors. The planned nominal position r_i^p of each vehicle is shown with an asterisk; its actual position r_i, marked with a disc of the same color, is always inside the computed robust tracking set $\tilde{\mathcal{X}}_i(r_i^p)$. *Source:* [119] *(reproduced with permission).*

(e.g. control effort, energy consumption, or sensor coverage) while enforcing that it never leaves the (time-varying) reach-avoid set.

In the standard implementation of STP (for both Algorithms 4.1 and 4.2), we have been choosing to initiate the trajectory at the latest viable departure time, in order to ensure minimum travel time. This means that the vehicle will start on the boundary of the reach-avoid set (as can be seen for vehicle 3 in the last plot of Figures 4.3 and 4.6). We then need to plan an optimal, minimum-time nominal trajectory as dictated by the Hamilton-Jacobi analysis, which will "ride" this time-varying boundary until it reaches the target set at exactly the scheduled time of arrival. If we were to deviate from such an optimal plan, our nominal trajectory would fall out of the reach-avoid set and by definition no longer be able to safely reach the target on time.

This restriction is relaxed if we initiate the nominal trajectory *earlier* than the latest viable departure time, allowing the backward-time propagation of the reach-avoid set to

continue, engulfing the starting state in its interior. Rather than needing to stay on the boundary of the reach-avoid set, then, the planned trajectory will have some flexibility to navigate its interior and still reach the target at (or possibly before) the scheduled time. A variety of different trajectory optimization tools can be used to plan an *efficient* trajectory, under the desirable metric, that remains inside the reach-avoid set. Crucially, while optimality of the solution will generally depend on the optimization algorithm of choice, feasibility is always guaranteed: any trajectory that attempts to exit the reach-avoid set can readily be converted into one that, upon reaching its boundary, switches to the minimum-time strategy dictated by the Hamilton-Jacobi analysis and safely reaches the target at the final time.

The Hamilton-Jacobi solution therefore acts as a last-resort enforcement of the safety and timeliness requirement: it will only intervene to the extent required to ensure that the nominal trajectory planned by each vehicle satisfies the problem specifications. Least-restrictive supervisory control schemes will be discussed extensively in Chapter 6, since they are a central component of the proposed framework for incorporating learning mechanisms into robotic systems' behavior while reliably ensuring safety.

4.5 Chapter Summary

This chapter has built on the time-varying reach-avoid game formulation introduced in Chapter 3 to propose a multi-agent trajectory planning framework that combines rigorous safety guarantees, minimum flight time properties, and highly scalable computation. The proposed Sequential Trajectory Planning (STP) scheme is motivated by the first-come-first-served flight plan allocation criterion adopted by NASA and the FAA in their Unmanned Aircraft System Traffic Management effort [6]. Thanks to the ability to handle time-varying state constraints, STP progresses through the set of vehicles in descending order of priority, representing the declared trajectories of all higher-priority vehicles as moving obstacles in the state space of each new vehicle. The resulting trajectory is not only collision-free, but is also guaranteed to reach the vehicle's goal set by the scheduled arrival time with the latest possible departure time subject to the already allocated higher-priority trajectories. In this sense, the STP scheme produces trajectories that are both safe and time-optimal subject to the priority ordering.

Notably, while the the general multi-agent trajectory planning problem is intrinsically combinatorial, the priority structure and the novel time-varying Hamilton-Jacobi reach-avoid analysis enable this sequential computation to overall scale linearly with the number of vehicles. The chapter discusses how the constraints induced by higher-priority vehicles should be recursively computed as the scheme progresses down the priority queue to ensure that each vehicle only requires a constant amount of computation.

Since runtime execution of the planned trajectories by the physical system will not generally be feasible without some amount of tracking error, the chapter also introduces a robust variant of the STP scheme, which reserves some amount of control authority at planning time for disturbance rejection at tracking time. Through an additional Hamilton-Jacobi

computation, which can be run ahead of time, it is then possible to ensure robust tracking of *any* trajectory produced by the STP scheme: provided that the dynamic disturbance term (capturing exogenous perturbations and modeling inaccuracies) remains within certain conservative bounds, the trajectory tracking error will also be bounded. Since this *tracking error bound* is independent of the actual trajectory plan, STP can easily account for it in its trajectory computations.

The simulation results presented in the chapter showcase the standard and robust variants of the STP scheme, demonstrating its correctness and scalability, in addition to the efficiency of the emerging trajectory patterns between large numbers of vehicles.

Finally, in some cases it may be desirable to replace the minimum-time criterion with a different efficiency objective, while still ensuring that all vehicles will safely reach their destinations by their scheduled times of arrival. The STP scheme can easily be converted into a *least-restrictive* supervisory law, which allows a different trajectory optimization algorithm to propose an efficient trajectory under more general criteria (energy, comfort, etc.), while ensuring safety and timeliness, preventing any proposed trajectory from abandoning the reach-avoid set. More details on using Hamilton-Jacobi analysis in a *least-restrictive* scheme will be discussed in Chapters 5 and 6.

Chapter 5

Safe Real-Time Robotic Navigation

> A good solution applied with vigor now is better than a perfect solution applied ten minutes later.
>
> George S. Patton (1885–1945)
> U.S. Army general

This chapter is based on the papers "Planning, Fast and Slow: A Framework for Adaptive Real-Time Safe Trajectory Planning" [16] and "Safely Probabilistically Complete Real-Time Planning and Exploration in Unknown Environments" [17], written in collaboration with David Fridovich-Keil, Sylvia Herbert, Sampada Deglurkar, and Claire Tomlin.

The ability to plan trajectories quickly and effectively is at the core of a robotic system's ability to successfully perform tasks in complex or uncertain environments. As we have seen in Chapter 4, it is possible to decouple the problem of tracking a nominal trajectory with a bounded error and planning a safe trajectory given this tracking error bound. In many practical cases, rather than plan nominal trajectories ahead of time, as in the UAS traffic management context, we may want to be able to generate real-time nominal trajectories through a given environment. Real-time planning (and replanning) is particularly important in cases where the environment is only partially known in advance, requiring adaptive exploratory behavior, or if there are dynamic obstacles or unknown agents whose future motion is not certain.[1]

In order to achieve real-time planning, it is common to use simplified dynamic models (in some cases purely kinematic models) enabling more tractable search or optimization.

[1] We will defer the treatment of uncertain moving obstacles until Chapter 7. For now, we note that uncertain moving objects can be handled under a worst-case analysis by considering their maximal (time-varying) forward-reachable set as a moving (typically growing) obstacle. Results are often overly conservative, motivating the need to replan in real time as the uncertainty gradually resolves itself.

Some of the most common approaches in this space are sampling-based planners such as rapidly-exploring random trees (RRTs) [121, 122], graph search planners such as A∗ [115], and combined approaches such as probabilistic roadmaps (PRMs) [123]. Since these approaches typically produce trajectories for a simplified model of the system, a key challenge lies in ensuring exact or more often approximate dynamic feasibility of the resulting plans when tracked by the actual physical system, in particular regarding satisfaction of safety constraints. The problem that we focus on in this section is that of providing robust tracking guarantees around such "fast" trajectory plans.

A common practice in robotics, illustrated in Figure 5.1 is to use a feedback controller based on a higher-order dynamical model (unsuitable for planning full trajectories in real time) to have the physical system locally and approximately track the low-order motion plan; this plan will in turn leave some "margin" around obstacles, in the hopes that the physical system will also avoid collisions in spite of tracking error. Without a bound on tracking error, however, the margin needs to be chosen heuristically: in some cases it may be overly conservative, unnecessarily restricting motion; in others, it may be insufficient, leading to constraint violations.

This chapter introduces a generalization of the *tracking error bound* concept from Chapter 4, Section 4.3 that enables the use of a simplified dynamical model for real-time planning, while retaining robust guarantees on the trajectory executed by the physical system. Section 5.1 lays down the mathematical formulation and presents a demonstration both in simulation and on a physical quadrotor platform. Section 5.2 then considers the problem of *recursive feasibility*, which is nontrivial under many important types of dynamics—in particular, if the robot or vehicle at hand is not able to quickly come to a stop or backtrack its previous trajectory, then careful additional analysis is needed to ensure that it will not collide or "get stuck" during exploration. Most commonly used motion planning schemes in robotics are meant for computational (rather than physical) environment exploration under static environment information, and as a result do not attempt to provide recursive feasibility guarantees. Nonetheless, we show in Section 5.2 that many of them can be augmented through the proposed scheme to retain safety and liveness properties through physical exploration with dynamically acquired environment information.

Related Work

There is an extensive body of literature in robot navigation and motion planning, which we cannot hope to fully summarize here. Rather, we focus on two main categories of closely related work and discuss several of the most relevant approaches.

Safe motion planning

Some modern motion planning approaches developed concurrently with the work presented in this chapter [124, 125] leverage a similar notion of a robust envelope around planned trajectories, which the physical system is guaranteed to remain within. Rather than Hamilton-

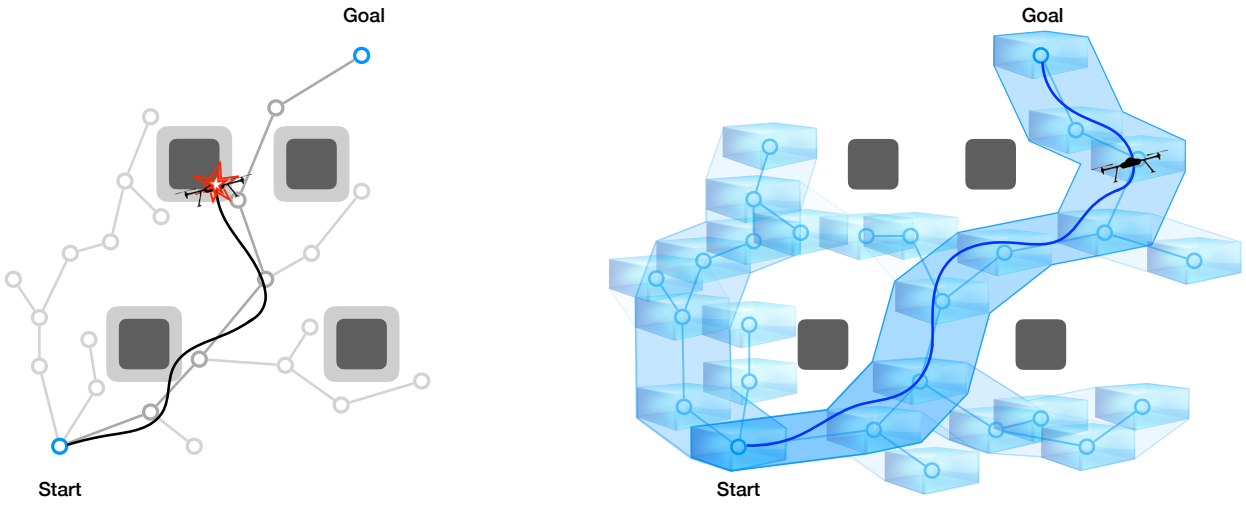


Figure 5.1: Illustration of heuristic-margin motion planning and the FaSTrack scheme. *Left:* A dynamical system may not be able to accurately track the output of a motion planner that assumes simplified dynamics. To mitigate this, planning algorithms often augment obstacles by a heuristic margin (light gray); however, the system trajectory may still deviate from the motion plan by more than this margin, leading to collision with an obstacle. *Right:* The FaSTrack scheme uses offline Hamilton-Jacobi analysis to compute a robust tracking error bound for a high-fidelity dynamical model, and a control policy that enforces it. Provided that discrepancies between the high-fidelity dynamics and the physical system's evolution are bounded by the specified disturbance set, the physical vehicle will track the motion plan safely.

Jacobi analysis, these other techniques rely on libraries of motion primitives and contraction theory respectively. A rigorous comparison between all of these methods is pending; since their structure introduces conservativeness in different forms, their relative advantages will likely be specific to the problem at hand.

Importantly, robust planning does not in itself guarantee recursive feasibility: a trajectory that is initially deemed to be robustly safe for a finite time horizon, or with limited environment information, may later become unsafe once the horizon is extended or new obstacles are detected. Richards and How [126] and Rosolia and Borrelli [127] directly address this problem within a model predictive control framework. The major differences between these works and our own are that [126] assumes linear time-invariant system dynamics, while [127] addresses an iterative task. Moreover, both assume *a priori* knowledge of all obstacles. Schouwenaars et. al. [128] also plan in a receding horizon, but as in our work,

recursive safety (though not liveness) is guaranteed by ensuring that all planned trajectories terminate in a safe loiter pattern.

The work in Section 5.2 may also be viewed as an extension of graph-based kinodynamic planners, e.g. the probabilistic roadmap [123], by enforcing that all edges in the graph are part of safely executable trajectories. Importantly, the framework introduced in this section guarantees recursive feasibility in an *a priori* unknown environment, with potentially high-order system dynamics, and in the presence of environmental disturbances.

Safe exploration

There is a rich body of work in robotic exploration methods, which tackle the problem of finding viable trajectories to a specified goal in an initially unknown environment. The majority of proposed methods, such as frontier-exploration [129–131] and D* [132, 133], have traditionally operated in configuration space, assuming a *kinematic* model of the robot's motion. Our method, in contrast, focuses on robotic systems for which a *dynamic* model is necessary, such as autonomous cars and aircraft. A sampling-based strategy is presented in [134] which reasons about inevitable collision sets [135], but is restricted to work with drift-less dynamics. More recent work [136] also addresses the dynamic exploration problem, but assumes that the vehicle is able to come to a stop in a short time.

Exploration has also been studied within the context of Markov Decision Processes (MDPs) and Reinforcement Learning (RL). Moldovan and Abbeel [137] propose an approach for generating a sequence of actions which preserve ergodicity with high probability. Other similar approaches, e.g. [138], also design risk-aware control policies that satisfy approximate constraints. Berkenkamp et. al. [139] and Chow et. al. [140] define safety in terms of Lyapunov stability. Though generally desirable, stability is insufficient to guarantee collision avoidance. The formulation of safe exploration in Section 5.2 is closely related to reachable sets and therefore provides a natural segue to the general safe learning framework introduced in Chapter 6.

5.1 Fast Planning, Safe Tracking

5.1.1 The FaSTrack Framework

The FaSTrack framework, initially introduced in [141], generalizes the tracking error bound and robust tracking set used in robust STP to the setting where planning and tracking are carried out under different dynamical models. Therefore, rather than a fully unstructured model error fully captured by the additive disturbance term, here we have a known structured discrepancy between the simplified model used for planning and the higher-fidelity model used for tracking. Since even the higher-fidelity model will not perfectly capture the evolution of the physical system, we continue to consider a bounded disturbance term to account for unmodeled dynamics.

Like robust STP, FaSTrack has an offline precomputation phase and an online planning phase. First, the offline precomputation phase determines the robust tracking error bound, together with a control policy to enforce it. Crucially, however, in addition to the disturbance rejection considered in robust STP, FaSTrack handles model discrepancy between planning and tracking. That is, the goal of the FaSTrack precomputation phase is to ensure that the high-fidelity system dynamics can robustly track any trajectory generated by a *different*, typically lower-order, dynamical model.

Once this robust tracking certificate is computed, the planning phase can be executed one or multiple times, in this case through a real-time motion planning algorithm. While in STP we considered trajectory plans obtained from Hamilton-Jacobi reach-avoid analysis, the FaSTrack framework does not assume a concrete trajectory planning method, provided that the class of trajectories it produces can be interpreted as (Carathéodory) solutions to some dynamical equation. For example, a geometric planner produces continuous (not necessarily smooth) paths in some space $\mathbb{R}^{n_r}$, which can be interpreted as constant-speed trajectories of a massless point with *simple motion* dynamics $\dot{x} = u$.

In this section, we assume that the environment can contain static *a priori* unknown obstacles provided they can be observed by the system within a certain sensing range. Assuming that the dynamical model used by the planner can instantaneously come to a full stop (as is the case for all geometric planners by setting $u = 0$), the minimum allowable sensing range from any state is equal to the circumscribed diameter of the robust tracking set. The stopping assumption is too restrictive in many cases, and we will consider more general recursive feasibility properties in Section 5.2.

The FaSTrack framework explicitly exposes a tradeoff between planning cost and conservativeness of plans. The higher the fidelity of the planning model, and thus the greater its similarity to the high-fidelity tracking model, the smaller the bound on the tracking error and therefore the less conservative motion plans will need to be to ensure safety at tracking time. On the other hand, higher-fidelity models are more challenging to plan with in real time (motivating the need for the FaSTrack scheme in the first place). As motion planning algorithms and on-board computation continue to improve in the coming years, the modeling gap may become reduced, leading to less conservative and more efficient motion plans, while still ensuring their safe execution by the physical system.

Finally, we recall the relation between the theoretical guarantee (for the high-fidelity dynamical model) and the practical assurance (for the physical system). The mathematical properties of the FaSTrack scheme guarantee that the high-fidelity model, including any dynamical system whose evolution can be expressed by some realization of the bounded disturbance input, will track the planned trajectory with a bounded error and without violating the specified constraints. This translates into an assurance about the physical system: to the extent that the evolution of the physical system is captured by the robust model, the properties asserted about the model also apply to the physical system. The contrapositive is also important: if the physical system violates any of the theoretical properties of the robust model, it must be because its behavior was not captured by the model.

5.1.2 Tracking and Planning Models

The *tracking model* should be a high-fidelity representation of the physical system's dynamics, and in general may present nonlinear, high-order dynamical equations. Let x represent the state variables of the tracking model, governed by dynamics

$$\dot{x} = f(x, u, d) \ , \tag{5.1}$$

with $x \in \mathbb{R}^n$, $u \in \mathcal{U}$, and $d \in \mathcal{D}$. Under the usual technical assumptions (Chapter 2, Section 2.1), trajectories (in Carathéodory's extended sense) are well-defined.

Example 5.1. *As a running example, we will consider a tracking model of a point mass (double-integrator) system with state $x = (r, v) \in \mathbb{R}^{2n_r}$, control input $u = a \in \mathcal{U} \subset \mathbb{R}^{n_r}$, and disturbance input $d = (d_v, d_a) \in \mathcal{D} \subset \mathbb{R}^{2n_r}$:*

$$\begin{bmatrix} \dot{r} \\ \dot{v} \end{bmatrix} = \begin{bmatrix} v - d_v \\ a - d_a \end{bmatrix} \tag{5.2}$$

The *planning model* defines the class of trajectories generated by the motion planner. Let x^p represent the state variables of the planning model. FaSTrack is agnostic to the type of planner, as long as its trajectories can be represented as (Carathéodory) solutions of a dynamical equation of the form

$$\dot{x^p} = f^p(x^p, u^p) \ , \tag{5.3}$$

with $x^p \in \mathbb{R}^{n^p}$ and $u^p \in \mathcal{U}^p$.

Example 5.2. *Our example planning dynamics will correspond to a geometric planner in $\mathbb{R}^{n_r}$, which can be interpreted as a massless point with state $x^p = r^p \in \mathbb{R}^{n_r}$ moving under a direct velocity control input $u^p = v^p \in \mathcal{U}^p \mathbb{R}^{n_r}$:*

$$\dot{r^p} = v^p \ . \tag{5.4}$$

Note that the planning model does not need a disturbance input. The trajectory resulting from the planning dynamics is the ideal, nominal reference that the physical vehicle will attempt to track, and as such, it is not affected by disturbances. Conversely, the tracking dynamics are meant to *approximately* represent the behavior of the physical system (which is what we are truly concerned with); since even a carefully crafted high-fidelity model is ultimately a mathematical abstraction, it can never fully capture the behavior of the physical system, and therefore we account for discrepancies with a bounded disturbance term.

In practice, it will typically be the case that $n^p \leq n$, and oftentimes the planning state variables will be a subset of the tracking state variables (i.e. the planning dynamics are a reduced-order version of the tracking dynamics). More generally, however, we need only assume that the target and obstacles are specified in terms of some shared state property

$r \in \mathbb{R}^{n_r}$ between the representations x^p and x, with $n, n^p \geq n_r$. This common property typically encodes the location or the pose of the system in the environment, so we will generally refer to $\mathbb{R}^{n_r}$ as the *configuration space*. We then have

$$\begin{aligned} \mathcal{T} &:= \{x \in \mathbb{R}^n : r \in \mathcal{T}^r\} \ , \qquad & \mathcal{T}^p &:= \{x^p \in \mathbb{R}^{n^p} : r^p \in \mathcal{T}^r\} \ , \\ \mathcal{F} &:= \{x \in \mathbb{R}^n : r \in \mathcal{F}^r\} \ , \qquad & \mathcal{F}^p &:= \{x^p \in \mathbb{R}^{n^p} : r^p \in \mathcal{F}^r\} \ , \end{aligned} \tag{5.5}$$

for some $\mathcal{T}^r, \mathcal{F}^r \subset \mathbb{R}^{n_r}$. As usual the dependence $r(x)$ and $r^p(x^p)$ is left implicit.

5.1.3 Error Dynamics

Similar to the robust STP analysis in Chapter 4, we want to study the error dynamics between the tracking and planning models, and seek to make this analysis independent of the absolute state of either system. We will assume the existence of a (possibly nonlinear) algebraic transformation $\phi : \mathbb{R}^n \times \mathbb{R}^{n^p} \to \mathbb{R}^n$ such that the tracking state x can always be obtained from the planning state x^p as a function of an additional*error* variable $e \in \mathbb{R}^n$,

$$x = \phi(e; x^p) \ , \tag{5.6}$$

such that $\phi(\cdot; x^p)$ is a bijection between e and x, and e dynamics are independent of x, x^p:

$$\dot{e} = f^e(e, u_e, u_e^p, d_e) \ , \tag{5.7}$$

with transformed inputs $u_e \in \mathcal{U}_e, u_e^p \in \mathcal{U}_e^p, d_e \in \mathcal{D}_e$ characterized through analogous mappings

$$u = \phi_u(u_e; x^p) \ , \qquad u^p = \phi_{u^p}(u_e^p; x^p) \ , \qquad d = \phi_d(d_e; x^p) \ , \tag{5.8}$$

which are also bijective for each fixed x^p. The corresponding input sets $\mathcal{U}_{e,i}, \mathcal{U}_{e,i}^p, \mathcal{D}_{e,i}$ (obtained by appropriately transforming $\mathcal{U}, \mathcal{U}^p, \mathcal{D}$) are assumed independent of x^p. Such an error representation often exists in practice, since the planning and tracking models are different abstractions of the same dynamical system, and a broad class of vehicle dynamics present desirable pose-invariance properties when expressed in the vehicle's body frame.

Example 5.3. *In our simple running example, the desired error representation can be obtained by subtracting the planning position states from the tracker's:*

$$x = \phi(e; x^p) = \begin{bmatrix} r^p + e_r \\ e_v \end{bmatrix} \ . \tag{5.9}$$

The error dynamics are then given by

$$\dot{e} = \begin{bmatrix} \dot{r} - \dot{r^p} \\ \dot{v} \end{bmatrix} = \begin{bmatrix} e_v + d_v - v^p \\ a + d_a \end{bmatrix} \ , \tag{5.10}$$

which, as we can see, depend solely on the error state e and inputs a, v^p, d_v, d_a.

5.1.4 Offline Hamilton-Jacobi Analysis

Analogously to robust STP, we can now solve a safety problem to determine a robust controlled invariant set $\Omega^{\mathcal{E}} \subset \mathbb{R}^n$ that the error e can be made to remain in for any nominal trajectory generated by the planner and for any realization of the uncertain tracking dynamics. In particular, we seek a robust controlled invariant set that will minimize the amount of tracking error in the dimensions that are relevant to the safety and completion specifications, in this case the *configuration space* $\mathbb{R}^{n_r}$. Setting the implicit surface function $l^{\mathcal{E}}(e) := -\|e_r\|_2$, our safety value function has the form of (4.29) and is the infinite-horizon limit of the unique viscosity solution to (4.30).

In principle, rather than choosing an arbitrary error radius $R^{\mathcal{E}}$ and verifying that the corresponding safe set is non-empty, we could directly determine the smallest non-empty superlevel set of the value function, which gives the tightest enforceable error bound. In practice, such a procedure suffers from numerical issues due to limited precision of the computed value function, so we usually choose a slightly larger radius $R^{\mathcal{E}}$ and use its corresponding safe set $\Omega^{\mathcal{E}}$, obtained as

$$\Omega^{\mathcal{E}} = \{e \in \mathbb{R}^n : V^{\mathcal{E}}(e) \geq -R^{\mathcal{E}}\} \ . \tag{5.11}$$

Example 5.4. *In our simple running example, letting the input sets $\mathcal{U}, \mathcal{U}^p, \mathcal{D}$ be decoupled in each of the n_r dimensions, the safety problem also becomes decoupled into n_r 2-dimensional subsystems. Figure 5.2 shows the tracking error bound $\Omega^{\mathcal{E}}$ for one subsystem.*

In this special case, an analytic solution can also be determined from the equations of motion, since the optimal error trajectories always correspond to constant (maximum) acceleration. Letting $\mathcal{U} = [-\bar{a}, \bar{a}]$, $\mathcal{U}^p = [-\bar{v^p}, \bar{v^p}]$, $\mathcal{D} = [-\bar{d}_v, \bar{d}_v] \times [-\bar{d}_a, \bar{d}_a]$ for this particular subsystem, the safe set is delimited by the two parabolas

$$\begin{aligned} e_r &= \frac{\frac{1}{2}\big(e_v - (\bar{v^p} + \bar{d}_v)\big)^2 - (\bar{v^p} + \bar{d}_v)^2}{\bar{a} - \bar{d}_a} \ , \\ e_r &= \frac{-\frac{1}{2}\big(e_v + (\bar{v^p} + \bar{d}_v)\big)^2 + (\bar{v^p} + \bar{d}_v)^2}{\bar{a} - \bar{d}_a} \ , \end{aligned} \tag{5.12}$$

corresponding to the maximum-acceleration characteristics (optimal trajectories) followed by the error state when the planning system switches abruptly between $\pm\bar{v^p}$ and d acts against the tracker's efforts to match velocity as quickly as possible to stop the drift in relative position. The analytic controlled invariant set is superimposed in Figure 5.2.

The Hamilton-Jacobi reachability precomputation is done in *in free space*, in other words, the error dynamics are constructed under the assumption that both the planning state and tracking state remain clear of their respective failure sets as given by $\mathcal{F}, \mathcal{F}^p$. This assumption

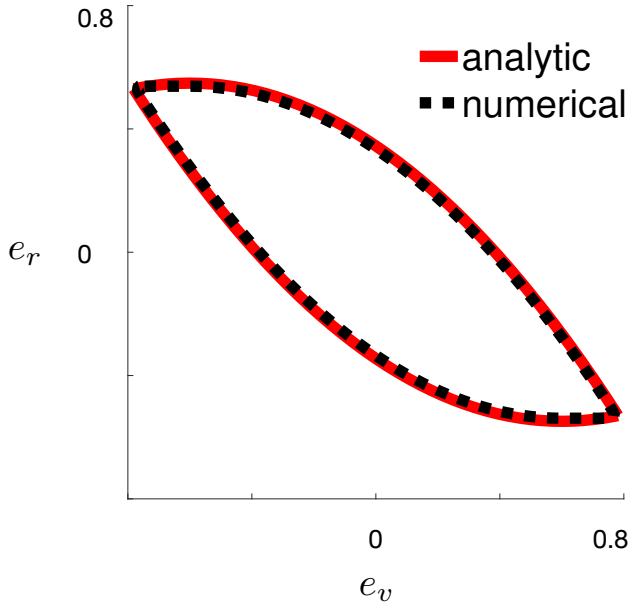


Figure 5.2: Analytically and numerically determined tracking error bound $\Omega^{\mathcal{E}}$, where the error e can be robustly kept despite worst-case disturbance and motion plan.

is not problematic, and will always hold when executing any planned trajectory output by the FaSTrack online planning phase.

5.1.5 Online Robust Trajectory Planning and Tracking

In the online phase, a motion planning algorithm is used to generate robustly safe nominal trajectories, which are then tracked by through a robust control policy derived from the Hamilton-Jacobi analysis.

Robust Real-Time Planning

The planning step, executed once at the start of the motion and again whenever new information about the environment is obtained through exploration, uses an off-the-shelf motion planning algorithm to generate a new nominal trajectory. The planning algorithm will implement a collision-checking routine, which in the FaSTrack scheme will be equipped with the precomputed tracking error bound.

The collision-checking operation requires translating the precomputed tracking error bound $\Omega^{\mathcal{E}} \subset \mathbb{R}^n$ into a space relevant to the planning problem. Through the transformation ϕ, we can determine the robust tracking set for any candidate planning state x^p, that is, the set of all possible states x in which the tracking system may find itself when attempting to track x:

$$\tilde{\mathcal{X}}(x^p) := \{\phi(e; x^p),\ e \in \Omega^{\mathcal{E}}\}\ . \tag{5.13}$$

However, since the state spaces of the tracking and planning systems are different, the set $\tilde{\mathcal{X}}(x^p)$ contains "spurious" dimensions that are not needed by the planning algorithm. Instead, we only need the *projection* of $\tilde{\mathcal{X}}(x^p)$ onto the configuration space $\mathbb{R}^{n_r}$, leading to the *projected robust tracking set*

$$\tilde{\mathcal{X}}^r(x^p) := \{r \in \mathbb{R}^{n_r} : \ r = r(x), \ x = \phi(e; x^p), \ e \in \Omega^{\mathcal{E}}\} \ . \tag{5.14}$$

For each candidate tracking statex^p, the projected robust tracking set $\tilde{\mathcal{X}}^r(x^p)$ is determined and checked for collisions. The specific collision-checking implementation depends on the environment representation used by the off-the-shelf planning algorithm. Since $\tilde{\mathcal{X}}^r(x^p)$ is typically encoded numerically on a discrete grid, collision checking may proceed by checking whether each cell in $\tilde{\mathcal{X}}^r(x^p)$ is occupied by an obstacle. This approach can often be made efficient by standard collision checking techniques, such as bounding $\tilde{\mathcal{X}}^r(x^p)$ by the sphere of radius $R^{\mathcal{E}}$ centered around r^p and discarding obstacles that do not intersect it, since by construction

$$\tilde{\mathcal{X}}^r(x^p) \subseteq \bar{B}(r^p, R^{\mathcal{E}})\} \ , \tag{5.15}$$

with $\bar{B}$ representing the closed ball. A faster collision-checking can also be implemented, at the cost of increased conservativeness, by replacing $\tilde{\mathcal{X}}^r(x^p)$ by this sphere altogether.

Example 5.5. *In our running example, noting that the configuration space is identical to the planning space, and substituting* (5.9) *and* (5.12) *into* (5.13)*, we obtain the projected robust tracking set*

$$\begin{aligned}\tilde{\mathcal{X}}^r(r^p) &= \{r \in \mathbb{R}^{n_r} : \ r = r(x), \ x = (r^p + e_r, e_v), \ (e_r, e_v) \in \Omega^{\mathcal{E}}\} \\ &= \prod_{j=1}^{n_r} \left[r_j^p - \frac{(\bar{v^p}_j + \bar{d}_{v,j})^2}{\bar{a}_j - \bar{d}_{a,j}}, r_j^p + \frac{(\bar{v^p}_j + \bar{d}_{v,j})^2}{\bar{a}_j - \bar{d}_{a,j}} \right] \ ,\end{aligned} \tag{5.16}$$

which is a Cartesian product of intervals in each dimension $j = 1, \ldots, n_r$ defining a box centered around r^p in $\mathbb{R}^m$. This is a substantially simplified form thanks to the fact that (a) the planning model is a lower-order abstraction of the tracking model, with the common relevant properties r being identical to the planning states x^p, and (b) the transformation ϕ between error and tracking state is independent of the planning state. Note that while (a) was trivially met in Section 4.3, (b) was not satisfied by the vehicle dynamics in (4.41)*. For many relevant systems we will need to use the more general form in* (5.13)*.*

Robust Tracking Policy

The tracking system may then apply the precomputed safety controller to track this planned trajectory in real time. The offline precomputation can be used to define a *least-restrictive*

control law, since the optimal tracking control enforcing the tracking error bound $\Omega^{\mathcal{E}}$ need only be applied on (or in practice in the vicinity of) the boundary $\partial\Omega^{\mathcal{E}}$.

Analogous to STP, the optimal tracking control can be determined in a straightforward manner through a look-up operation in real time. At any given time t, the controller first computes the current error $e = \phi^{-1}\big(\mathbf{x}(t); \mathbf{x}^p(t)\big)$. The transformed optimal control $\pi_e^{\mathcal{E}}(e)$ can be obtained by applying (4.32), either through an online computation of the numerical gradient $\nabla_e V^{\mathcal{E}}$ or by looking up $\pi_e^{\mathcal{E}}(e)$ in a previously stored table. The optimal tracking control $\pi^{\mathcal{E}}(e) = \phi_u\big(\pi_e^{\mathcal{E}}(e); \mathbf{x}^p(t)\big)$ can then be applied to enforce the tracking error bound. Continually doing this over time is guaranteed to keep the tracking trajectory $\mathbf{x}(\tau)$ within the tracking error set $\tilde{\mathcal{X}}(\mathbf{x}^p(\tau)$ throughout the motion plan.

Whenever replanning takes place, the new nominal trajectory $\mathbf{x}_{k+1}^p$ is set to start at a future planned state along the current nominal trajectory $\mathbf{x}_k^p$, that is: $\mathbf{x}_{k+1}^p(t+\delta) := \mathbf{x}_k^p(t+\delta)$. The time margin $\delta > 0$ ensures that the planning algorithm has enough time to compute the new plan and update the nominal trajectory for the vehicle to track.

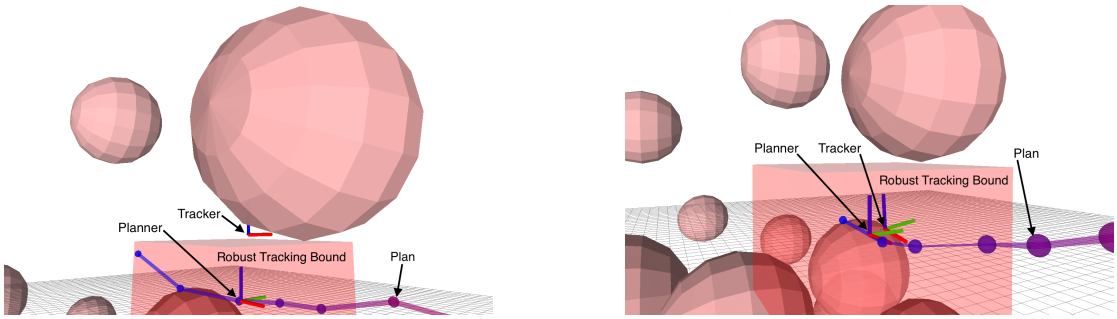
5.1.6 Meta-Planning: Motion Plans with Multiple Models

As we have seen, the FaSTrack framework presents system designers with a tradeoff in the choice of planning model: simpler planning models can be used to quickly compute nominal trajectories, but they will typically be harder to track accurately, leading to larger tracking error bounds and forcing these nominal trajectories to be more conservative. Similarly, planning models allowing more aggressive maneuvers (e.g. kinematic models assuming faster motion) will seem to allow the robot to cover more space in less time, but may also lead to larger tracking errors, due to the limited maneuverability of the high-fidelity tracking model.

It is possible to at least partly automate this tradeoff by allowing designers to specify a suite of planning models, each of them possibly associated with a different motion planning algorithm; we refer to each model-algorithm pair as a *planner*. As with the tracking and planning model, the suite of planning models must share a *configuration space* of state variables relevant to trajectory safety and completion. The planning phase can then explore the space through a *meta-planning* scheme that selects different planners as it constructs candidate trajectories. The resulting nominal trajectory output by this scheme may then consist of multiple sub-trajectories "stitched" together in the configuration space $\mathbb{R}^{n_r}$.

Through this process, trajectory planning can make use of the simplest or most maneuverable planners in relatively unconstrained of the environment where large tracking errors are not problematic, and automatically switch to higher-fidelity models or more conservative maneuverability constraints in more cluttered regions where tighter tracking guarantees may be necessary to ensure safety. These switches are mediated automatically by the *meta-planning* scheme, which can be given a preference for using more efficient models where possible.

A planner switch involves a change from one tracking error set to another. Ensuring that this transition is done safely, and providing a bound on possible tracking errors reached during the transition, requires an additional Hamilton-Jacobi precomputation for every al-



(a) Standard LQR controller. (b) Safety controller.

Figure 5.3: Simulated autonomous flight in a cluttered environment. Using a standard feedback controller for real-time trajectory tracking, such as LQR, the quadrotor may leave the projected robust tracking set in the vicinity of obstacles. In contrast, the optimal safety controller enforces the bound. Video: `https://youtu.be/lPdXtR8Ar-E`

lowable planner-to-planner transition. We direct the interested reader to [16] for a detailed description of one possible sample-based *meta-planning* implementation for a suite of geometric motion planners with different velocities.

5.1.7 FaSTrack Implementation Demonstration

We implemented the FaSTrack framework within the robot operating system (ROS) [142] framework. Code is written in C++ and is available as an open source ROS package.[2] We additionally implemented a *meta-planning* extension of FaSTrack, which automatically chooses among a finite collection of planning models as it constructs the nominal trajectory, adjusting the chosen model to the local environment constraints.[3] The code can be used in conjunction with the Open Motion Planning Library (OMPL) [143].

Here, we demonstrate the FaSTrack framework on a 6D near-hover quadrotor model tracking a 3D geometric planning model in the cluttered environment depicted in Figure 5.3. Planned trajectories are determined running the BIT* algorithm [144]. The dynamics for the tracking model are given by

$$\begin{bmatrix} \dot{\mathrm{x}} \\ \dot{\mathrm{y}} \\ \dot{\mathrm{z}} \\ \dot{v}_{\mathrm{x}} \\ \dot{v}_{\mathrm{y}} \\ \dot{v}_{\mathrm{z}} \end{bmatrix} = \begin{bmatrix} v_{\mathrm{x}} - d_{v_{\mathrm{x}}} \\ v_{\mathrm{y}} - d_{v_{\mathrm{y}}} \\ v_{\mathrm{z}} - d_{v_{\mathrm{z}}} \\ g \tan\theta - d_{a_{\mathrm{x}}} \\ -g \tan\phi - d_{a_{\mathrm{y}}} \\ T - g - d_{a_{\mathrm{z}}} \end{bmatrix}, \tag{5.17}$$

[2] `https://github.com/HJReachability/fastrack`
[3] `https://github.com/HJReachability/meta_fastrack`

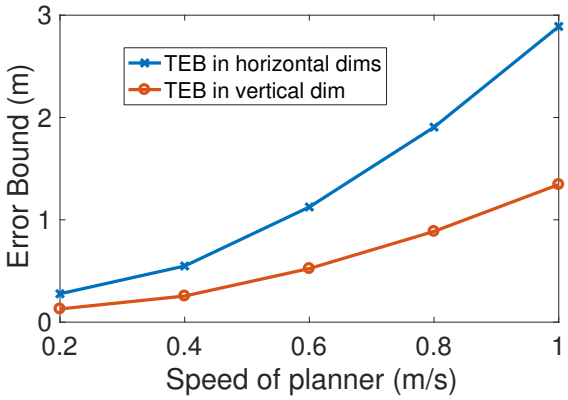


Figure 5.4: Robust tracking bound size vs. speed assumed by the planner in each subsystem.

where the tracking control $u = (\theta, \phi, \tau)$ corresponds to roll, pitch, and thrust. In all experiments, we set $\theta, \phi \in [-0.15 \text{ rad}, 0.15 \text{ rad}]$ and $\tau \in [7.81 \text{ m/s}^2, 11.81 \text{ m/s}^2]$. Note that we have assumed a zero yaw angle for the quadrotor. This is enforced in practice by using a feedback controller on yaw rate to regulate yaw to zero.

The *simple motion* dynamics for the planning model are

$$\begin{bmatrix} \dot{\mathrm{x}}^p \\ \dot{\mathrm{y}}^p \\ \dot{\mathrm{z}}^p \end{bmatrix} = \begin{bmatrix} v_{\mathrm{x}}^p \\ v_{\mathrm{y}}^p \\ v_{\mathrm{z}}^p \end{bmatrix} \tag{5.18}$$

where $u^p = [v_{\mathrm{x}}^p, v_{\mathrm{y}}^p, v_{\mathrm{z}}^p]$ represent the planning model's velocity control in each dimension. We will be considering a suite of such planners with different maximum speeds in each dimension: 1.0 m/s, 0.7 m/s, 0.4 m/s and 0.1 m/s.

Choosing the error representation $e = (e_r, e_v)$, $e_r = (r - r^p)$, $e_v = v$, the error dynamics between the tracking and planning models are:

$$\begin{bmatrix} \dot{e}_{\mathrm{x}} \\ \dot{e}_{\mathrm{y}} \\ \dot{e}_{\mathrm{z}} \\ \dot{e}_{v_{\mathrm{x}}} \\ \dot{e}_{v_{\mathrm{y}}} \\ \dot{e}_{v_{\mathrm{z}}} \end{bmatrix} = \begin{bmatrix} v_{\mathrm{x}} - d_{v_{\mathrm{x}}} - v_{\mathrm{x}}^p \\ v_{\mathrm{y}} - d_{v_{\mathrm{y}}} - v_{\mathrm{y}}^p \\ v_{\mathrm{z}} - d_{v_{\mathrm{z}}} - v_{\mathrm{z}}^p \\ g \tan\theta - d_{a_{\mathrm{x}}} \\ -g \tan\phi - d_{a_{\mathrm{y}}} \\ T - g - d_{a_{\mathrm{z}}} \end{bmatrix} \tag{5.19}$$

Equation (5.19) can be split into three 2D subsystems with states $(x, v_x), (y, v_y)$, and (z, v_z). Note that the dynamics of the $(\mathrm{x}, v_{\mathrm{x}})$ and $(\mathrm{y}, v_{\mathrm{y}})$ subsystems are identical, and thus can be solved once and applied to each subsystem. Decomposing the Hamilton-Jacobi reachability analysis as in [145], we compute the $(\mathrm{x}, v_{\mathrm{x}})$ set in 2 min 15 s and the $(\mathrm{z}, v_{\mathrm{z}})$ set in 2 min, for a total of a 4 min 15 s precomputation time. Since the tracker dynamics of each subsystem can be transformed into the double-integrator form in Example 5.1, and the planner dynamics are already in the *simple motion* form of Example 5.2, we can equivalently

compute the tracking error bound and optimal bound-enforcing controller in closed form, following the analysis in Examples 5.3, 5.4, and 5.5. Figure 5.4 shows the growth of the projected robust tracking set $\tilde{\mathcal{X}}^r$ in each subsystem's position state as the planner speed in that dimension increases.

Due to the form of (5.17), the optimal safety controller will be *bang-bang*, that is, control variables will always be assigned extreme values within the allowed ranges, possibly alternating between maximum and minimum; however, recall that it is only critical to apply the safety control near the *boundary* of the tracking error bound $\Omega^{\mathcal{E}}$. A smooth linear controller may be used in the interior, following a *least-restrictive* supervisory control scheme.

Simulation

Figure 5.3 shows a snapshot of a simulated autonomous quadrotor flight in an artificial environment with spherical obstacles using trajectories generated by our algorithm. Initially, the obstacle locations and sizes are unknown to the quadrotor, but as soon as they come within the sensing radius (which in the case of a geometric planner must be no less than the circumscribed diameter of the robust tracking set) they are added to the meta-planner's internal environment model and used during re-planning.

Figure 5.3a demonstrates an undesirable outcome obtained when tracking a nominal trajectory with a standard LQR controller; in Figure 5.3b everything remains the same except that we apply the optimal controller derived from the offline analysis in Section 5.1.4. Note that the LQR controller makes no guarantee regarding the robust tracking set, and indeed leaves the set in the vicinity of the obstacle. The optimal controller, conversely, is guaranteed to keep the system in the robust tracking set.

Hardware Demonstration

We replicated the simulation on a hardware testbed using the Crazyflie 2.0 open source quadrotor platform, shown in Figure 5.5. We obtained position and orientation measurements at ~ 235 Hz from an OptiTrack infrared motion capture system. Given state estimates, we send control signals over a radio to the quadrotor at 100 Hz. As shown in the summary video available online, the quadrotor successfully avoids the obstacles while remaining inside the robust tracking set for each planner the meta-plan. Trajectory planning (including *meta-planning*) typically runs in well under one second in a moderately cluttered environment.

Figure 5.6 shows the quadrotor's position over time recorded during a hardware demonstration. Note that the quadrotor stays well within the projected robust tracking set throughout the flight. As seen in Figure 5.7, the *meta-planning* scheme ensures this even as the robust tracking bound changes size during planner switches.

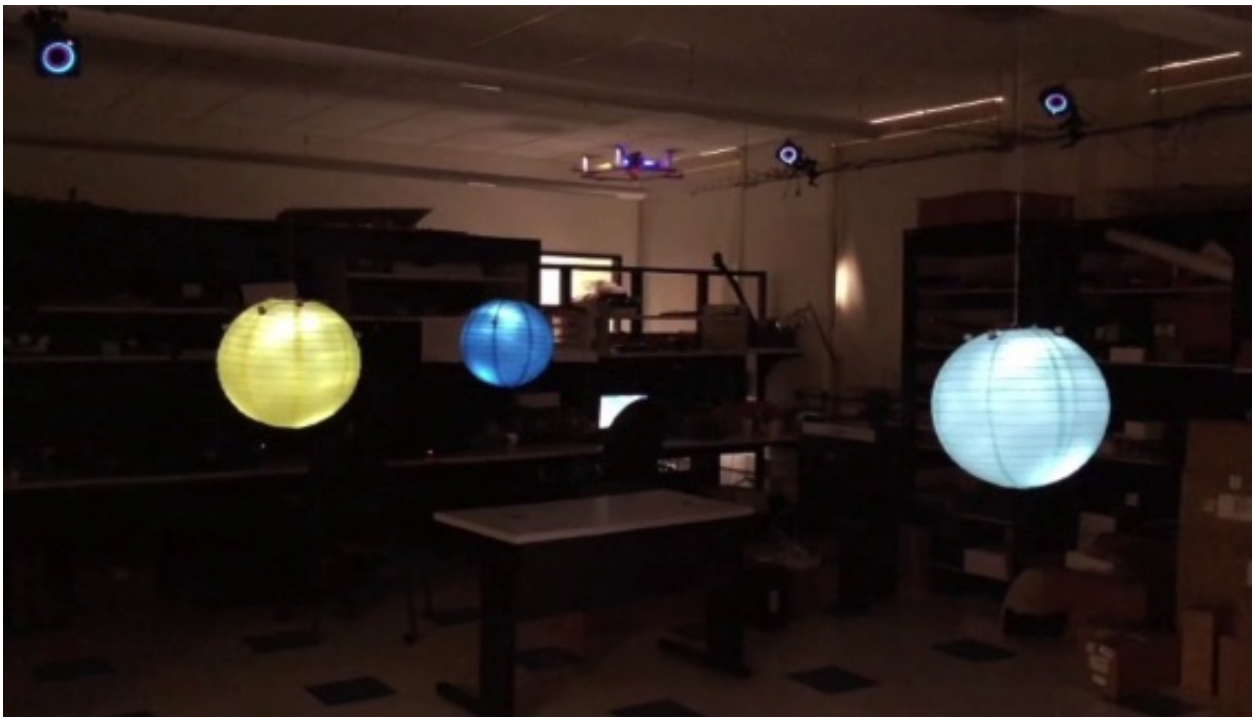


Figure 5.5: A Crazyflie 2.0 flying during our hardware demonstration. Three OptiTrack motion capture cameras are visible in the background. Obstacles, marked by lanterns, are revealed to the planner based on proximity (simulating a sensor).

5.2 Recursive Safety and Liveness in Uncertain Environments

The analysis introduced in Chapter 4 and Section 5.1 allows us to compute trajectories for one or multiple robots or autonomous vehicles while providing assurances about the physical system's ability to safely track the planned trajectories in spite of discrepancies between the system's dynamics and the model used for motion planning. In Section 5.1, we demonstrated the functioning of this framework for real-time planning (and replanning) as the system explores an *a priori* unknown environment. Provided that a feasible motion plan *could always be found* when executing the online planning step, the higher-fidelity tracking dynamics were then guaranteed to robustly track the associated trajectory while avoiding constraint violations. In the examples and demonstrations considered, it was always possible to find such a feasible plan, since a geometric planner always has the trivial option to stop in place and avoid collisions for all future times.

More generally however, this condition is nontrivial, and can pose a central challenge to safe exploration of *a priori* unknown environments. In many cases of interest, especially involving vehicles and mobile robots, it is not realistic to assume that the dynamical system (or any reasonably descriptive approximation) can instantaneously come to a stop; if obsta-

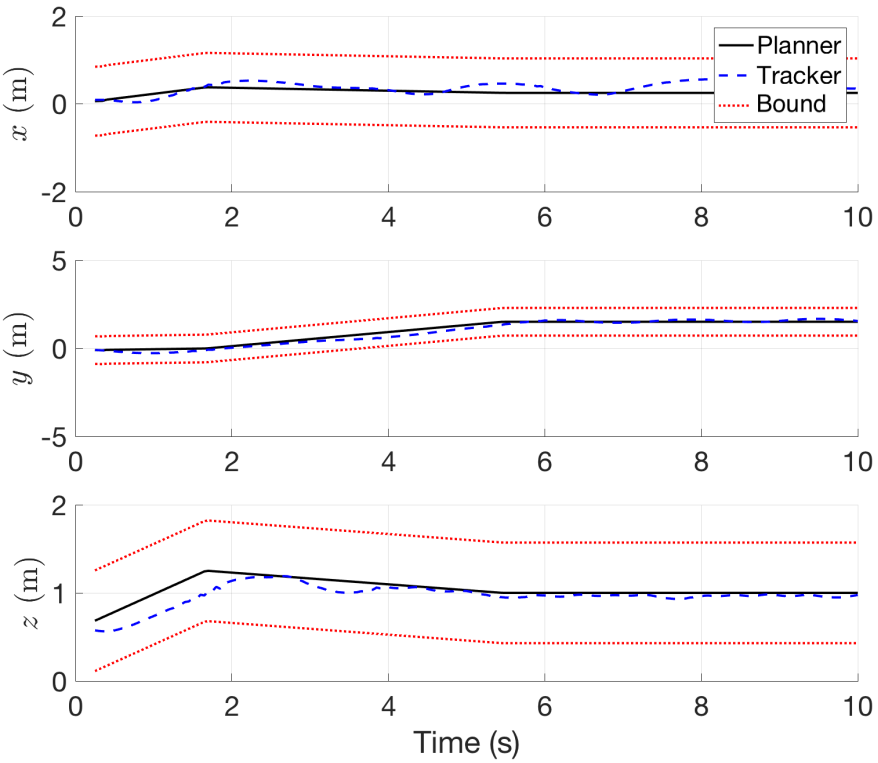


Figure 5.6: Quadrotor trajectory in FaSTrack hardware demonstration. The physical vehicle remains within the tracking bound around the nominal trajectory plan at all times.

cles need to be sensed online, it can be difficult to guarantee *recursive feasibility*. Informally, a planning algorithm is recursively feasible if it explores the environment without losing its ability to maintain safety or to reach the goal. The dangers of non-recursively feasible exploration are illustrated in Figure 5.8: given a limited sensing range or planning horizon, a planning scheme that only ensures constraint satisfaction for a finite time window may unwittingly lead the system into states from which satisfying the constraints is no longer possible, i.e. *unsafe* states, or from which the Many commonly-used motion planning formulations bypass these issues, for example by assuming full prior knowledge of the environment or by assuming, as we did in Section 5.1, that it is safe to stop and possible to do so either instantaneously or within a short range. While such assumptions are acceptable in many scenarios, there are important applications and systems for which ensuring safe dynamic exploration and navigation becomes crucial. This is particularly true for a wide range of vehicles, such as motorcycles, automobiles, and fixed-wing aircraft, which may operate at high speeds and cannot easily (if at all) come to a stop. These issues are especially pronounced for non-holonomic systems.

In this section, we consider the process of incrementally building a graph of forward-reachable states (for a given planning model) within known free space, while simultaneously identifying those states from which the initial state is safely reachable. This latter graph

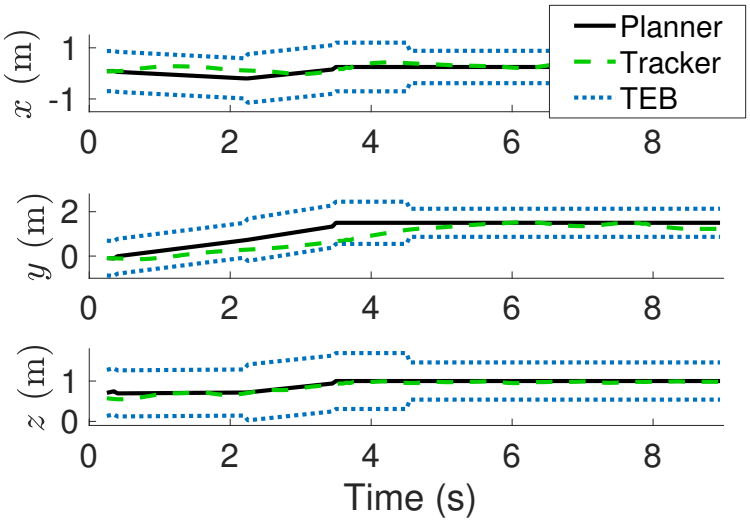


Figure 5.7: Quadrotor trajectory in FaSTrack hardware demonstration with *meta-planning*. Even though the tracking bound varies in size as the aggressiveness of the nominal trajectory plan changes, the physical vehicle always remains within the bound.

implicitly represents a discrete under-approximation of the backward-reachable set of the initial state. Restricting the physical exploration of the system to only consider nominal trajectories from which either the initial state or the target are safely reachable then allows us to provide all-time assurances for the physical system, through the same analysis introduced in this chapter. More specifically, this procedure ensures:

- *Safety:* all trajectories initiated by the physical system will be robustly collision-free.
- *Liveness:* if the target is safely reachable from the initial state, it will always be safely reachable.
- *Safe Probabilistic Completeness:* if a target was originally reachable by a plan that preserves the ability to return home, the proposed exploration process guarantees that it will eventually be found with probability 1.

5.2.1 Recursive Feasibility: Safety and Liveness

We consider a bounded environment $\mathcal{W} \subset \mathbb{R}^{n_r}$, where $\mathbb{R}^{n_r}$ is the *configuration space* of state properties relevant to the specification of targets and constraints (recall that this typically includes position or pose configuration, although it may generally comprise any state variables). We denote by $\mathcal{P}$ the set of planning states whose corresponding configuration lies in $\mathcal{W}$.

There is an initially unknown (or possibly partially known) set $\mathcal{F}^r$ containing all configurations that, when adopted by the system, result in a failure state (for example, a physical

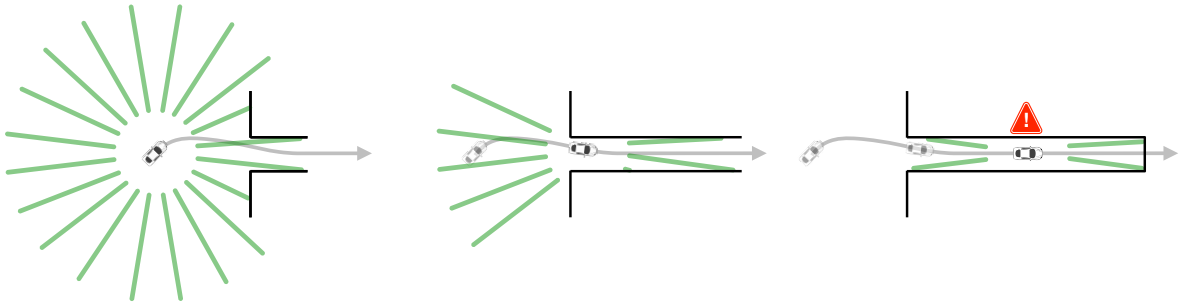


Figure 5.8: Schematic illustration of an unsafe motion plan due to the lack of recursive feasibility. The planner initially guides the vehicle into an opening between two obstacles, following a trajectory that is *temporarily* safe given the finite sensing field of view (here depicted by the green rays). Upon following the trajectory, the vehicle's sensors detect a dead end; at this point, the planning algorithm is unable to find a new trajectory that would avoid a collision.

collision), as per (5.5). In addition, the goal of the navigation problem is to eventually guide the system's state into a target state, also defined, following (5.5), in terms of the configuration space, $\mathcal{T}^r \subset \mathbb{R}^{n_r}$. The motion planning formulation in Chapter 4 and Section 5.1, we had assumed that targets are known *a priori*. Since we are presently interested in exploratory navigation problems, we relax this assumption here. As with $\mathcal{F}^r$, then, the system may have full, partial, or no initial knowledge of $\mathcal{T}^r$.

We assume that for each configuration $r \in \mathcal{W}$, the environment representation can assign a label $\{\text{OCCUPIED}, \text{FREE}, \text{GOAL}, \text{UNKNOWN}\}$. The system's knowledge of the environment will be updated online according to measurements from a well-characterized sensor, with field of view given by a set-valued map $\mathcal{V} : \mathbb{R}^n \rightrightarrows \mathcal{W}$. We will restrict our attention to deterministic sensing models, i.e. if a previously UNKNOWN configuration $r \in \mathcal{W}$ comes within the sensor's field of view $\mathcal{V}(x)$, it will be immediately identified as OCCUPIED if $r \in \mathcal{F}^r$, and otherwise labeled as GOAL if $r \in \mathcal{T}^r$ and FREE otherwise. Probabilistic extensions are possible, though beyond the scope of our analysis here.

After completing the offline Hamilton-Jacobi analysis introduced in Section 5.1.4, the projected robust tracking set $\tilde{\mathcal{X}}^r(x^p) \subset \mathbb{R}^{n_r}$ can be determined for any candidate planning state $x^p \in \mathcal{P}$, following (5.14). Using this set, a candidate planning state x^p can be collision-checked against any known OCCUPIED configurations as well as all (potentially occupied) UNKNOWN configurations.

We will require that the system is at all times guaranteed to only take configurations known to be FREE (and, eventually, GOAL). For convenience, we will denote by $\mathcal{W}_{\text{free}}(t_0)$ the set of configurations $r \in \mathcal{W}$ that are labeled as FREE or GOAL at any particular time t_0. By construction, $\mathcal{W}_{\text{free}}(t_0) \subseteq \mathcal{K}^r$, where the constraint set $\mathcal{K}^r$ is the complement of the failure set $\mathcal{F}^r$ in $\mathcal{W}$.

Definition 5.1 (Robustly Collision-Free State)**.** *A nominal planning state $x^p \in \mathcal{P}$ is robustly*

collision-free given the environment information if all configurations in its corresponding projected robust tracking set are labeled FREE or GOAL:

$$\tilde{\mathcal{X}}^r(x^p) \subseteq \mathcal{W}_{\text{free}}(t_0) \ . \tag{5.20}$$

We denote by $\mathcal{P}_{\text{free}}(t_0)$ the set of known robustly collision-free planning states at time t_0.

In addition, a planning state x^p will be deemed a *goal* state if every configuration in its projected robust tracking set $\tilde{\mathcal{X}}^r(x^p)$ is labeled as GOAL (in which case it is guaranteed that the tracking system's state will reach $\mathcal{T}$ given a viable planning trajectory to x^p). We will denote by $\mathcal{W}_{\text{goal}}(t_0)$ the set of configurations $r \in \mathcal{W}$ that are labeled as GOAL at time t_0. By construction, $\mathcal{W}_{\text{goal}}(t_0) \subseteq \mathcal{W}_{\text{free}}(t_0)$ and in addition $\mathcal{W}_{\text{goal}}(t_0) \subseteq \mathcal{T}^r$.

Definition 5.2 (Known Goal State)**.** *A nominal planning state $x^p \in \mathcal{P}$ is a known goal state if all configurations in its corresponding projected robust tracking set are labeled GOAL:*

$$\tilde{\mathcal{X}}^r(x^p) \subseteq \mathcal{W}_{\text{goal}}(t_0) \ . \tag{5.21}$$

We denote by $\mathcal{P}_{\text{goal}}(t_0)$ the set of known goal states at time t_0.

By Definitions 5.1, and 5.2, all known goal states at time t_0 are also known to be robustly collision-free: $\mathcal{P}_{\text{goal}}(t_0) \subseteq \mathcal{P}_{\text{free}}(t_0)$. In addition, as in Section 4.3, we let the *robust target set* $\tilde{\mathcal{T}}^p$ (4.34) be given by all nominal planning states for which the robust tracking set is contained in the target set, or, in the *configuration space*,

$$\tilde{\mathcal{T}}^p := \{x^p \in \mathbb{R}^{n^p} : \tilde{\mathcal{X}}^r(x^p) \subseteq \mathcal{T}^r\} \ . \tag{5.22}$$

We then have that $\mathcal{P}_{\text{goal}}(t_0) \subseteq \tilde{\mathcal{T}}^p$ for all times t_0, that is, all known goal states belong to the robust target set for the planning system. In fact, the two sets become equal whenever the system has full knowledge of the target: with all configurations in $\mathcal{T}^r$ labeled GOAL, we have $\mathcal{W}_{\text{goal}}(t_0) = \mathcal{T}^r$ and therefore the definitions of $\mathcal{P}_{\text{goal}}(t_0)$ and $\tilde{\mathcal{T}}^p$ become equivalent.

Definition 5.3 (Known Safe Trajectory)**.** *A planned trajectory $\mathbf{x}^p\big(\cdot; t_0, x_0^p, \mathbf{u}^p\big)$ is known at time t_0 to be safe if its states are robustly collision-free for all time based on the environment information available at t_0:*

$$\forall t \geq t_0, \tilde{\mathcal{X}}^r\Big(\mathbf{x}^p\big(t; t_0, x_0^p, \mathbf{u}^p\big)\Big) \subseteq \mathcal{W}_{\text{free}}(t_0) \ . \tag{5.23}$$

Definition 5.4 (Known Safe Reachable Set)**.** *The known safe forward-reachable set $\mathcal{R}_F$ of a set of states $\mathcal{P}' \subseteq \mathcal{P}$ at time t_0 is the set of states $x^p \in \mathcal{P}$ that can be reached from $\mathcal{P}'$ by a trajectory $\mathbf{x}^p$ known at t_0 to be safe:*

$$\begin{aligned} \mathcal{R}_F(\mathcal{P}'; t_0) := \Big\{ x^p \mid \exists x^{p\prime} \in \mathcal{P}', \exists t \geq t_0, \exists \mathbf{u}^p, \forall \tau \in [t_0, t] : \\ \tilde{\mathcal{X}}^r\Big(\mathbf{x}^p\big(\tau; t_0, x^{p\prime}, \mathbf{u}^p\big)\Big) \subseteq \mathcal{W}_{\text{free}}(t_0), x^p = \mathbf{x}^p\big(t; t_0, x^{p\prime}, \mathbf{u}^p\big) \Big\} . \end{aligned} \tag{5.24}$$

Analogously, the known safe backward-reachable set $\mathcal{R}_B$ of $\mathcal{P}'$ at t_0 is the set of states $x^p \in \mathcal{P}$ from which $\mathcal{P}'$ can be reached by a trajectory $\mathbf{x}^p$ known at t_0 to be safe:

$$\mathcal{R}_B(\mathcal{P}'; t_0) := \Big\{ x^p \mid \exists x^{p\prime} \in \mathcal{P}', \exists t \geq t_0, \exists \mathbf{u}^p, \forall \tau \in [t_0, t] : \tilde{\mathcal{X}}^r\Big(\mathbf{x}^p\big(\tau; t_0, x^p, \mathbf{u}^p\big)\Big) \subseteq \mathcal{W}_{\text{free}}(t_0), x^{p\prime} = \mathbf{x}^p\big(t; t_0, x^p, \mathbf{u}^p\big) \Big\}. \tag{5.25}$$

We will often consider reachable sets of individual states; for conciseness, we will write $\mathcal{R}_B(x^p; t_0)$ rather than $\mathcal{R}_B(\{x^p\}, t_0)$.

Definition 5.5 (Viability)**.** *A state x^p is* viable *at time t_0 given a home state x^p_{home} if at t_0 it is known to be possible to safely reach either some known goal state $x^p_{\text{goal}} \in \mathcal{P}_{\text{goal}}(t_0)$ or x^p_{home} from x^p, i.e. $x^p \in \mathcal{R}_B\big(\mathcal{P}_{\text{goal}}(t_0) \cup \{x^p_{\text{home}}\}; t_0\big)$. A trajectory $\mathbf{x}^p$ is viable at t_0 if all states along $\mathbf{x}^p$ are viable at t_0.*

Note that a viable trajectory at time t_0 is necessarily a known safe trajectory at time t_0; the converse, however, is not generally true, that is, a trajectory can be known safe (Def. 5.3) but not viable. Consider for example a trajectory that enters an enclosed FREE region within which it is possible to maintain a collision-free loiter pattern indefinitely, but the region cannot be left without colliding. Such a trajectory is not viable, since it will render the system unable to continue its exploration. This leads us to ask what states may be explored without losing viability.

Definition 5.6 (Safely Explorable Set)**.** *Given an initial planning state x^p_{home}, the safely explorable set $\mathcal{P}_{\text{SE}} \subset \mathcal{P}$ is the collection of states that can eventually be visited by the system through a trajectory starting at state x^p_{home} with no prior knowledge of $\mathcal{W}$ whose states are, at each time $t \geq 0$, viable according to the known free space $\mathcal{W}_{\text{free}}(t)$.*

Based on the idea of the safely explorable set we can finally introduce the important notion of *liveness* for the purposes of our work.

Definition 5.7 (Liveness)**.** *Given a set $\mathcal{T}^r \subset \mathbb{R}^r$ of target configurations, a planning state x^p is live if it is possible to reach some $x^p_{\text{goal}} \in \tilde{\mathcal{T}}^p$ from x^p while preserving viability at all times, i.e if $\tilde{\mathcal{T}}^p \cap \mathcal{P}_{\text{SE}} \neq \emptyset$. A trajectory $\mathbf{x}^p$ is live if all states in $\mathbf{x}^p$ are live.*

Note that, unlike the previous definitions, Definitions 5.6 and 5.7 are not tied to the available knowledge of the environment at a given time t_0: rather, safe explorability and liveness are ground-truth properties, which can only be determined by the system after sufficient exploration of the environment. Typically, the system may start at a state that is live but not be initially able to determine this; liveness of a state or a trajectory can usually only be determined retrospectively once a safe trajectory to a goal state is computed.

Finally, we will refer to a planning algorithm as *recursively feasible* if, given that the initial state $x^p_0 =: x^p_{\text{home}}$ is live, all future states x^p output by the planner (and therefore tracked by the system) are both live and viable. We will show that the navigation scheme

presented here is recursively feasible; crucially, this means that it preserves liveness of the system's planning state even when the property cannot yet be determined based on existing environment information. Moreover, we will also show that the scheme is *safely probabilistically complete*, in the sense that, if x_0^p is live, then with probability 1 the planning state will eventually reach $\tilde{\mathcal{T}}^p$ (and therefore the tracking system will robustly reach $\mathcal{T}$) through continued guaranteed safe exploration.

5.2.2 The Recursively Feasible Navigation Framework

The framework introduced here is comprised of two concurrent, asynchronous operations: building a graph of states which discretely under-approximate the forward and backward-reachable sets of the initial "home" state, and traversing this graph to find recursively feasible trajectories. Namely, we define the graph $\mathcal{G}_F := \{V, E\}$ of vertices V and edges E. Vertices are individual states in $\mathcal{P}$, and directed edges are trajectories $\mathbf{x}^p$ between pairs of vertices. $\mathcal{G}_F$ will be a discrete under-approximation of the currently known safe forward-reachable set of the initial state x^p_{home}. We also define the graph $\mathcal{G}_B \subseteq \mathcal{G}_F$ to contain only those vertices in $\mathcal{G}_F$ which are in the known safe backward-reachable set of $\{x^p_{\text{home}}\} \cup \mathcal{P}_{\text{goal}}(t_0)$, and the corresponding edges. We use the notation $x^p \in \mathcal{G}_F$ to mean that state x^p is a vertex in $\mathcal{G}_F$, and likewise for $\mathcal{G}_B$.

We use following two facts extensively. They follow directly from the definitions above and our assumptions on deterministic sensing and a static environment.

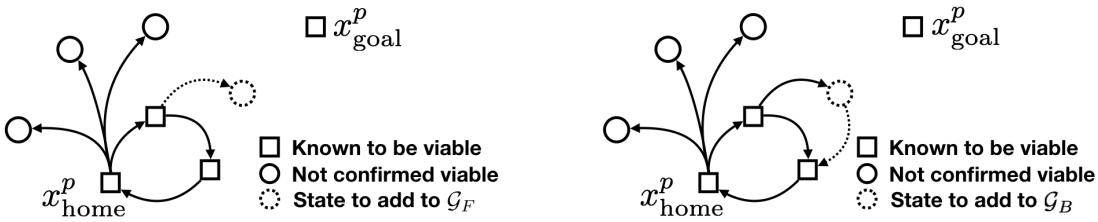
Lemma 5.1 (Permanence of Known Safety)**.** *A trajectory $\mathbf{x}^p$ that is known to be safe at time t_0 will continue to be known safe at all $t \geq t_0$.*

Proof. Since sensing is assumed to be deterministic, configurations $r \in \mathcal{W}$ that have been labeled as FREE will not be re-labeled. Therefore, for all t_0, t with $t_0 \leq t$, $\mathcal{W}_{\text{free}}(t_0) \subseteq \mathcal{W}_{\text{free}}(t)$ and by Definition 5.3, known safety of $\mathbf{x}^p$ at time t_0 implies known safety of $\mathbf{x}^p$ at all future times t. □

Lemma 5.2 (Permanence of Known Safe Reachability)**.** *A state x^p that is in the known safe forward– or backward-reachable set of some $\mathcal{P}_0 \subset \mathcal{P}$ at time t_0 will continue to belong to this set for all $t \geq t_0$, i.e. $\mathcal{R}_F(\mathcal{P}_0; t_0) \subseteq \mathcal{R}_F(\mathcal{P}_0; t)$ and $\mathcal{R}_B(\mathcal{P}_0; t_0) \subseteq \mathcal{R}_B(\mathcal{P}_0; t)$.*

Proof. By assumption, we have a known safe forward (backward) trajectory connecting some $x_0^p \in \mathcal{P}_0$ and x^p that is safe at t_0. From Lemma 5.1 this trajectory is also safe for all times $t \geq t_0$, which implies that x^p belongs to the known safe forward– (backward–) reachable set of x_0^p, and thereby of $\mathcal{P}_0$, at time t. □

Corollary 5.1 (Permanence of Known Safe Reachability of Non-Decreasing Set)**.** *If a state x^p is in the known safe forward– (backward–) reachable set of a set $\mathcal{P}_0 \subset \mathcal{P}$ at time t_0, it is also in the known safe forward– (backward–) reachable set of any set $\mathcal{P}_1 \supseteq \mathcal{P}_0$ for all $t \geq t_0$.*



(a) Outbound expansion. (b) Inbound consolidation.

Figure 5.9: In outbound expansion (a), a new state is sampled from $\mathcal{P}$ and added to $\mathcal{G}_F$ if safely reachable from $\mathcal{G}_F$. In inbound consolidation (b) a state in $\mathcal{G}_F$ is added to $\mathcal{G}_B$ if it can safely reach a (viable) state in $\mathcal{G}_B$.

Corollary 5.1 implies that any set in the known safe backward-reachable set of the known goal set $\mathcal{P}_{\text{goal}}(t_0)$ at time t_0 will also be in the known safe backward-reachable set of the updated known goal set $\mathcal{P}_{\text{goal}}(t) \supseteq \mathcal{P}_{\text{goal}}(t_0)$ at any future time $t \geq t_0$.

5.2.3 Building the Graph

We incrementally build the graph by alternating between outbound expansion and inbound consolidation steps. In the outbound expansion step, new candidate states are sampled, and if possible, connected to $\mathcal{G}_F$. This marks them as part of the forward-reachable set of x^p_{home}. In the inbound consolidation step, we attempt to find a safe trajectory from forward-reachable states in $\mathcal{G}_F$ back to a state in $\mathcal{G}_B$, which is known to be *viable*. Successful inbound consolidation marks a state as either able to reach $\mathcal{P}_{\text{goal}}(t)$ (and therefore $\tilde{\mathcal{T}}^p$) or safely return to x^p_{home}.

Outbound expansion

This process incrementally expands a discrete under-approximation $\mathcal{G}_F$ of the forward-reachable set of the home state, $\mathcal{R}_F(x^p_{\text{home}}; t)$. Note that, by Lemma 5.2, $\mathcal{R}_F(x^p_{\text{home}}; t)$ can only grow as the environment $\mathcal{W}$ is gradually explored over time and therefore any state x^p added to $\mathcal{G}_F$ at a given time t is guaranteed to belong to $\mathcal{R}_F(x^p_{\text{home}}; t')$ for all $t' \geq t$.

We add states to $\mathcal{G}_F$ via a Monte Carlo sampling strategy inspired by existing graph-based kinodynamic planners [116], illustrated in Figure 5.9a. We present a relatively simple strategy here, although more sophisticated options for sampling new states are possible, e.g. [144, 146].

Let x^p_{new} be sampled uniformly at random from $\mathcal{P}_{\text{free}}$ at time t. We wish to establish whether or not x^p_{new} is in the known safe forward-reachable set of home at t, i.e. $x^p_{\text{new}} \in \mathcal{R}_F(x^p_{\text{home}}; t)$. This is accomplished by invoking a third-party motion planner, which will attempt to find a safe trajectory to x^p_{new} from any of the points already known to be in $\mathcal{R}_F(x^p_{\text{home}}; t)$. In Section 5.2.6, we use a standard kinodynamic planner from the OMPL [143] for this purpose.

We observe that repeatedly executing this procedure will, in the limit, result in a dense discrete under-approximation of $\mathcal{R}_F(x^p_{\text{home}}; t)$. Formally, assuming that the low-level planner will find a valid trajectory to a sampled state x^p if one exists, then for any $\epsilon > 0$, we have that the probability that a new sampled state $x^{p\prime} \in \mathcal{R}_F(x^p_{\text{home}}, t)$ will lie within a distance of ϵ from the nearest state $x^p \in \mathcal{G}_F$ goes to 1 in the limit of infinite samples. We formalize this observation below:

Lemma 5.3 (Dense Sampling)**.** *For all $\epsilon > 0$, assuming we sample candidate states x^p uniformly and independently from $\mathcal{P}$ and $\mathcal{P}$ is compact, then letting x^p_k be the k-th sampled state from $\mathcal{P}$ we have that $\forall t$:*

$$\lim_{k\to\infty} P\big(\min_{x^p \in \mathcal{G}_F} \|x^p_k - x^p\| < \epsilon \mid x^p_k \in \mathcal{R}_F(x^p_{\text{home}}; t)\big) = 1 \ .$$

Proof. This follows directly from the properties of uniform sampling from compact sets. □

This will be useful in proving the safe probabilistic completeness of the recursive feasibility scheme.

Inbound consolidation

This process incrementally adds states in $\mathcal{G}_F$ to a discrete under-approximation $\mathcal{G}_B$ of the known safe *backward*-reachable set of $\mathcal{P}_{\text{goal}}(t) \cup \{x^p_{\text{home}}\}$. By Definition 5.5, any state added to this set is *viable*. As granted by Lemma 5.1 and Corollary 5.1, this also means that a trajectory will *always* exist from such a state to either x^p_{home} or some $x^p_{\text{goal}} \in \mathcal{P}_{\text{goal}}(t) \subseteq \tilde{\mathcal{T}}^p$. This is a crucial element of our overall guarantee of recursive feasibility. We recall that $\mathcal{G}_B \subseteq \mathcal{G}_F$ by construction.

Suppose that, at time t, $x^p \in \mathcal{G}_F \setminus \mathcal{G}_B$. We will attempt to add x^p to $\mathcal{G}_B$ by finding a safe trajectory from x^p to any of the states currently in $\mathcal{G}_B$ by invoking the low-level motion planner. If we succeed in finding such a trajectory, then by construction there exists a trajectory all the way to x^p_{home} or some $x^p_{\text{goal}} \in \mathcal{P}_{\text{goal}}(t)$, so we add x^p to $\mathcal{G}_B$. If x^p is added to $\mathcal{G}_B$, we also add all of its ancestors in $\mathcal{G}_F$ to $\mathcal{G}_B$, since there now exists a trajectory from each ancestor through x^p to either x^p_{home} or x^p_{goal}. This procedure is illustrated in Figure 5.9b.

5.2.4 Exploring the Graph

When requested, we must be able to supply a safe trajectory beginning at the current state reference $\mathbf{x}^p(t)$ tracked by the system. Recall from Section 5.1 that under the FaSTrack framework the physical system's state $x(t) \in \mathbb{R}^n$ is guaranteed to remain within a robust tracking set $\tilde{\mathcal{X}}(x^p)$. This property allows us to make guarantees in terms of planning model states x^p rather than full physical system states x.

Trajectories $\mathbf{x}^p$ output by the recursive feasibility scheme must guarantee future safety for all time; that is, as the system follows $\mathbf{x}^p$ we must always be able to find a safe trajectory starting from any future state. In addition, we require that x^p_{home} remains safely reachable

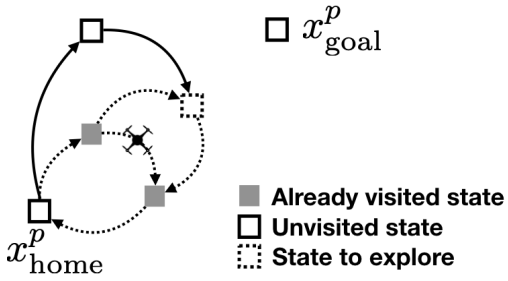


Figure 5.10: Schematic diagram of the heuristic exploration procedure.

throughout the trajectory; this ensures that *liveness* is preserved (if it was possible from x^p_{home} to safely explore $\mathcal{W}$ and reach $\mathcal{T}^r$ then this possibility will not be lost by embarking on $\mathbf{x}^p$). Liveness in this context is thus a critical property, complementary to safety.

By construction, any cycle in $\mathcal{G}_B$ will be known to be safe at all future times (Lemma 5.1). Readily, this suggests that we could guarantee perpetual recursive feasibility by always returning the same cycle. However, this naive strategy, while safety-preserving, would never reach the goal. Moreover, it would not incrementally explore the environment. In order to force the system to explore unknown regions of $\mathcal{W}$, we modify this naive strategy by routing the system through a randomly selected *unvisited* viable state $x^p_{\text{new}} \in \mathcal{G}_B$, and then back to x^p_{home}, as illustrated in Figure 5.10. The trajectory always ends in a periodic safe orbit between x^p_{new} and x^p_{home}. Note that this random selection does not need to be done naively (e.g. by uniform sampling of unvisited states in $\mathcal{G}_B$), and efficient exploration strategies are certainly possible. In our examples we will use an ϵ-greedy sampling heuristic by which, with probability $1-\epsilon$, we select the unvisited $x^p \in \mathcal{G}_B$ closest to $\mathcal{P}_{\text{goal}}(t)$ (if non-empty, otherwise we can select the unvisited $x^p \in \mathcal{G}_B$ farthest from x^p_{home}), and otherwise, with probability ϵ, we uniformly sample an unvisited state in $\mathcal{G}_B$. Other strategies are certainly possible, and there is room to make exploration efficient using metrics like expected exploration time and sensor coverage.

As soon a state $x^p_{\text{goal}} \in \mathcal{P}_{\text{goal}}(t)$ is added to $\mathcal{G}_B$ for the first time, we may simply return a trajectory from the current state $\mathbf{x}^p(t)$ to x^p_{goal}. This will always be possible because, by construction of output trajectory plans, $\mathbf{x}^p(t)$ belongs to an edge of the graph $\mathcal{G}_B$ and can therefore safely reach some node $x^p \in \mathcal{G}_B$. From x^p, a known safe trajectory exists that will guide the system to x^p_{goal} (if necessary, looping through x^p_{home}).

5.2.5 Algorithm Summary and Theoretical Guarantees

To summarize, the proposed navigation scheme maintains, at each time t, graph representations of the forward-reachable set of x^p_{home} and the backward-reachable set of $\mathcal{P}_{\text{goal}}(t) \cup \{x^p_{\text{home}}\}$. Over time, these graphs become increasingly dense (Lemma 5.3). Additionally, all output trajectories terminate at some $x^p_{\text{goal}} \in \tilde{\mathcal{T}}^p$ or in a cycle that includes x^p_{home}. This implies our main theoretical result:

Theorem 5.1 (Recursive Feasibility)**.** *Assuming that we are able to generate an initial viable trajectory (e.g. a loop through x^p_{home}), all subsequently generated trajectories will be viable and preserve the liveness of x^p_{home}. Thus, the proposed navigation scheme guarantees* recursive feasibility*.*

Proof. By assumption, the initial trajectory $\mathbf{x}^p_0$ output at t_0 is safe (Definition 5.3). We now proceed by induction: assume that the i-th reference trajectory $\mathbf{x}^p_i$ is viable for the knowledge of free space at the time t_i at which it was generated, i.e. $\forall t \geq t_i, \mathbf{x}^p_i(t) \in \mathcal{R}_B(\mathcal{P}_{\text{goal}}(t_i) \cup \{x^p_{\text{home}}\}; t_i)$. Assuming x^p_{goal} has not been reached yet at the time of the next planning request, t_{i+1}, a new trajectory is generated from initial state $\mathbf{x}^p_i(t_{i+1})$, after possibly updating the environment information, with $\mathcal{P}_{\text{free}}(t_{i+1}) \supseteq \mathcal{P}_{\text{free}}(t_i)$ and $\mathcal{P}_{\text{goal}}(t_{i+1}) \supseteq \mathcal{P}_{\text{goal}}(t_i)$. The new trajectory $\mathbf{x}^p_{i+1}$ is constructed by concatenating safe trajectories between states in $\mathcal{G}_B \subseteq \mathcal{R}_B(\mathcal{P}_{\text{goal}}(t_{i+1}) \cup \{x^p_{\text{home}}\}; t_i)$ and therefore is a viable trajectory. Such a trajectory can always be found, because it is always possible to choose $\mathbf{x}^p_{i+1} \equiv \mathbf{x}^p_i$, which, by the inductive hypothesis was a viable trajectory at time t_i and, by Lemma 5.2 and Corollary 5.1, continues to be viable at t_{i+1}. This means that from every state visited by $\mathbf{x}^p_{i+1}$ there is a known safe trajectory to either reach $\mathcal{P}_{\text{goal}}(t_i)$ or return to x^p_{home}. In the former case, $\mathbf{x}^p_{i+1}$ is in fact *live* (and, since $\forall t \geq 0, \mathbf{x}^p_{i+1}(t) \in \mathcal{R}_F(x^p_{\text{home}}; t_{i+1})$, x^p_{home} is retrospectively determined to be live as well); in the latter case, $\mathbf{x}^p_i$ will also inherit the (possibly not yet known) liveness of x^p_{home}, by observing that $\forall t \geq 0, \mathbf{x}^p_{i+1}(t) \in \mathcal{R}_B(x^p_{\text{home}}; t_{i+1})$. □

Corollary 5.2 (Dynamical System Exploration)**.** *Given that the safety and viability of trajectories is evaluated using the projected robust tracking set $\tilde{\mathcal{X}}^r(x^p)$, and the dynamical system's configuration $r(x)$ is guaranteed to remain within this set by the Hamilton-Jacobi analysis in Section 5.1, Theorem 5.1 implies that the dynamical system can continually execute safe trajectories in the environment and, if a nominal trajectory to the target is found, is guaranteed to safely reach the target by robustly tracking this trajectory.*

Moreover, we ensure that each output trajectory visits an unexplored state in $\mathcal{G}_B$, which implies that $\mathcal{G}_B$ approaches the safely explorable set $\mathcal{P}_{\text{SE}}$ from Definition 5.6. Together with Theorem 5.1, this implies the following completeness result:

Theorem 5.2 (Safe Probabilistic Completeness)**.** *In the limit of infinite runtime, the recursive feasibility scheme eventually finds a viable trajectory to the robust target set $\tilde{\mathcal{T}}^p$ with probability* 1 *if it is not disjoint from the safely explorable set.*

Proof. By Theorem 5.1, all trajectories output will be viable; hence, the autonomous system will remain safe for all time (Corollary 5.2). Further, since each generated trajectory visits a previously unvisited state in $\mathcal{G}_B$ with nonzero probability, by Lemma 5.3 it will eventually observe (and appropriately label) new regions in the safely explorable set $\mathcal{P}_{\text{SE}}$ if any exist. Moreover, those regions will eventually be sampled, added to $\mathcal{G}_B$, and visited by subsequent trajectories. Because we have assumed all sets of interest to be bounded, this implies that we will *almost surely* eventually identify and add a state $x^p_{\text{goal}} \in \tilde{\mathcal{T}}^p \cap \mathcal{P}_{\text{SE}} x^p_{\text{home}}$ to $\mathcal{G}_B$ if such a state exists (that is, if $\tilde{\mathcal{T}}^p$ and $\mathcal{P}_{\text{SE}} x^p_{\text{home}}$ are not disjoint). □

We conclude this section with several brief remarks regarding implementation.

In Section 5.2.3, we specify that calls to the planning algorithm should seek to connect states to existing states in $\mathcal{G}_F$ and $\mathcal{G}_B$. In practice, we find that connecting to one of the k nearest neighbors (measured in the Euclidean norm over $\mathcal{P}$) in the appropriate graph suffices.

In Section 5.2.4, we describe traversing $\mathcal{G}_B$ to find safe trajectories between vertices. For efficiency, we recommend keeping track of the following variables at each vertex: cost-from-home, cost-to-home, and cost-to-goal, where cost may be any consistent metric on trajectories (e.g. duration). If these quantities are maintained, then care must be taken to update them appropriately for descendants and ancestors of states that are added to $\mathcal{G}_F$ and $\mathcal{G}_B$ in Section 5.2.3.

Finally, we observe that outbound expansion, inbound consolidation, and graph exploration may all be performed in parallel and asynchronously.

5.2.6 Numerical Example

We demonstrate the recursive feasibility scheme in a real-time simulation, implemented within the Robot Operating System (ROS) software environment [142].

Let the high-order system dynamics be given by the following 6D model:

$$\dot{x} = \begin{bmatrix} \dot{x} \\ \dot{v}_x \\ \dot{y} \\ \dot{v}_y \\ \dot{z} \\ \dot{v}_z \end{bmatrix} = \begin{bmatrix} v_x \\ g\cos u_1 \\ v_y \\ -g\sin u_2 \\ v_z \\ u_3 - g \end{bmatrix} \tag{5.26}$$

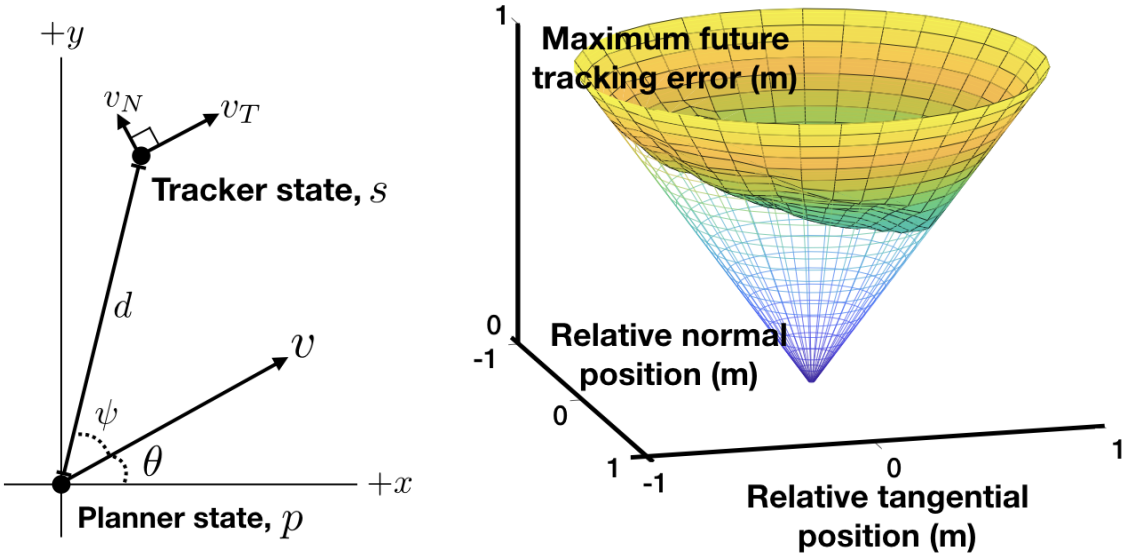
where g is acceleration due to gravity, the states are position and velocity in (x, y, z), and the controls are $u_1 = \text{pitch}, u_2 = \text{roll}$, and $u_3 = \text{thrust acceleration}$. These dynamics are a reasonably accurate model for a lightweight quadrotor operating near a hover and at zero yaw.

We consider the following lower-order 3D dynamical model for planning:

$$\dot{x^p} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} v\cos\theta \\ v\sin\theta \\ u^p \end{bmatrix} \tag{5.27}$$

where v is a constant tangential speed in the Frenet frame, states are absolute heading θ, and (x, y) position in fixed frame, and control u^p is the turning rate. We interpret these dynamics as a Dubins car operating at a fixed z height z.

We take controls to be bounded in all dimensions independently by known constants: $u \in [\underline{u}_1, \overline{u}_1] \times [\underline{u}_2, \overline{u}_2] \times [\underline{u}_3, \overline{u}_3]$ and $u^p \in [\underline{u^p}, \overline{u^p}]$. In order to compute the FaSTrack tracking error bound $\Omega^{\mathcal{E}}$, we solve the Hamilton-Jacobi safety problem (4.30) for the *error dynamics*



(a) Relative states. (b) Maximum-future-error function.

Figure 5.11: (a) Relative states for 6D near-hover quadrotor tracking 3D Dubins car. (b) Projected future error (min over e_{v_T}, e_{v_N}) for each (e_T, e_N) in the planner's frame.

defined by (5.26) and (5.27). In this case, the error dynamics are given by:

$$\dot{e} = \begin{bmatrix} \dot{\rho} \\ \dot{\psi} \\ \dot{v}_T \\ \dot{v}_N \end{bmatrix} = \begin{bmatrix} v_T \cos\psi + v_N \sin\psi \\ -u^p - v_T \sin\psi + v_N \cos\psi \\ u_1 \cos\theta - u_2 \sin\theta + u^p v_T \\ -u_1 \sin\theta - u_2 \cos\theta - u^p v_T \end{bmatrix} \tag{5.28}$$

with the error states ρ (distance), ψ (relative bearing), v_T (tangential velocity), and v_N (normal velocity) illustrated in Figure 5.11a.

Figure 5.11b is a projection of the robust tracking value function computed using numerical Hamilton-Jacobi analysis tools [53]. For better visualization, the plot shows the *negative* of the value function, which encodes the maximum future tracking error metric $-l^{\mathcal{E}}(e) := e_\rho$ from each initial error state. We use the over-approximation (5.15) for rapid collision-checking during each call to the low-level motion planner. Since the high-order dynamics (5.26) do allow for variation in z, we also incorporate a z dimension for $\Omega^{\mathcal{E}}$ which may be obtained by solving a similar differential game in the (z, v_z) subsystem of (5.26), as in [16].

We use the KPIECE1 kinodynamic planner [147] within the Open Motion Planning Library (OMPL) [143] to plan all trajectories for the low-level dynamics while building the graphs $\mathcal{G}_F$ and $\mathcal{G}_B$. We model static obstacles as spheres in $\mathbb{R}^3$ and use an omnidirectional sensing model in which all obstacles within a fixed range of the vehicle are sensed exactly. These choices of environment and sensing models are deliberately simplified in order to more

clearly showcase the recursive feasibility scheme. The framework itself is compatible with arbitrary representations of static obstacles and deterministic sensing models. Extensions to dynamic obstacles and probabilistic sensing are promising directions for future research.

We demonstrate the recursive feasibility scheme in the simulated environment shown in Figure 5.12, designed to illustrate the importance of maintaining recursive feasibility. This simulation is intended as a proof of concept; the central contribution of this section is theoretical and applies to a range of planning problems.

Observe in Figure 5.12 that the proposed navigation scheme avoids collision where a non-recursively-feasible exploration approach would often fail. Here, the target is directly in front of the home position, beyond what *appears* to be a potential passage between two obstacles. However, just beyond the sensor's field of view $\mathcal{V}$, there is a narrow dead end. Many standard planning techniques would either optimistically assume the unknown regions of the environment are free space, or plan in a receding horizon within known free space $\mathcal{W}_{\text{free}}(t)$. In both cases, the planner would tend to guide the system into the narrow dead end—especially if the target's location or general direction is known *a priori*—leading to a crash (recall that the planner's speed v is fixed).

By contrast, the recursive feasibility scheme takes a more circuitous—but recursively feasible—route to the target. The evolution of planned viable trajectories is shown on the right in Figure 5.12. Initially, it plans tight loops near x^p_{home}, but over time it visits a larger region of the safely explorable space $\mathcal{P}_{\text{SE}}$, and eventually finds (and executes) a viable trajectory to $x^p_{\text{goal}} \in \tilde{\mathcal{T}}^p$.

5.3 Chapter Summary

This chapter has extended the notion of robust tracking through a guaranteed tracking error bound to enable the computation of real-time plans with a simplified dynamical model different from the higher-fidelity model used for dynamical tracking analysis. In addition, an adaptive scheme is proposed that switches between multiple available motion planners, constructing a safely executable motion plan out of locally collision-free trajectories. The robustly safe real-time planning framework is demonstrated in simulation and on a physical quadrotor platform.

Building on the robust tracking guarantees, the chapter additionally introduces a scheme for recursively feasible motion planning and environment exploration. The approach combines the notions of forward and backward reachability to construct sample-based graphs in the planning state space that enable guaranteeing safety for all time, even for planning dynamics that cannot come to a stop, as demonstrated through a fixed-speed Dubins car numerical example. Moreover, the scheme guarantees that if the initial state is *live*, i.e. the goal is safely explorable from it, then the resulting motion plans will preserve liveness, and eventually (with probability 1) find a safe trajectory to the goal. To our knowledge, this is the first motion planning scheme to provide such a robust recursive feasibility guarantee regarding safety and liveness of exploration for a dynamical system.

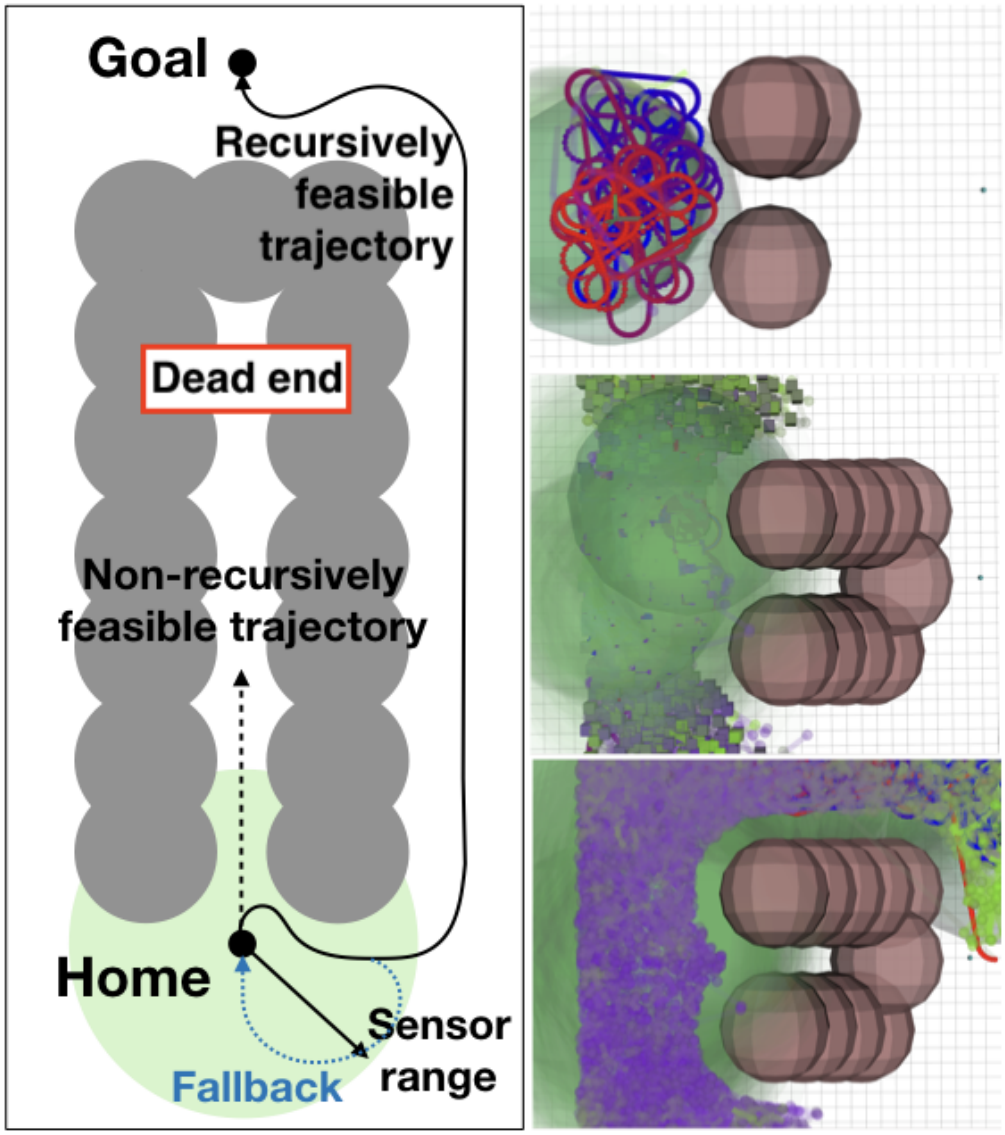


Figure 5.12: Recursively feasible exploration scheme using a Dubins car model with a fixed minimum turning radius and constant speed. *Left:* Schematic diagram of an environment in which a non-recursively feasible planning algorithm could enter a narrow dead end and fail to recover. *Right:* Snapshots of the recursive feasibility scheme over time. We build a search graph in known free space, identifying robustly viable trajectories that can safely return to the initial state or directly reach the goal. The robotic system incrementally explores the environment along these recursively feasible plans and is guaranteed, with probability 1, to eventually identify and traverse a viable trajectory to the goal, if one exists (bottom right). Video: `https://youtu.be/GKQwFxdJWSA`

Part II

Safety Across the Reality Gap

Chapter 6

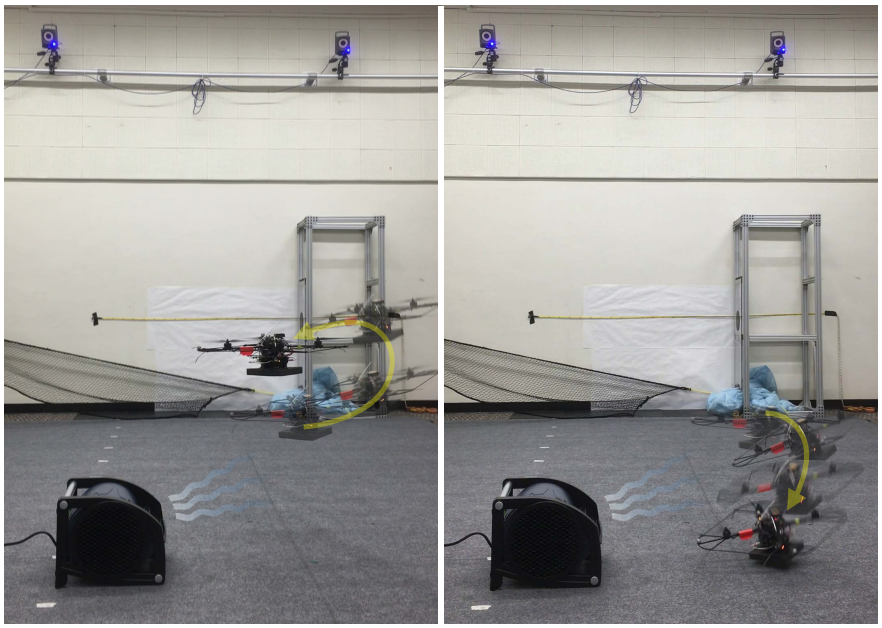
Safe Learning under Uncertainty

> One never seeks to avoid one trouble without running into another; but prudence consists in knowing how to distinguish the character of troubles, and for choice to take the lesser evil.
>
> Niccolò Machiavelli,
> *The Prince*, 1513

This chapter is based on the paper "A General Safety Framework for Learning-Based Control in Uncertain Robotic Systems" [18], written in collaboration with Kene Akametalu, Melanie Zeilinger, Shahab Kaynama, Jeremy Gillula, and Claire Tomlin.

Learning-based methods in control and artificial intelligence are generating a considerable amount of excitement in the research community. The auspicious results of deep reinforcement learning schemes in virtual environments such as arcade videogames [148] and physics simulators [149], make these techniques extremely attractive for robotics applications, in which complex dynamics and hard-to-model environments limit the effectiveness of purely model-based approaches. However, the difficulty of interpreting the inner workings of many machine learning algorithms (notably in the case of deep neural networks), makes it challenging to make meaningful statements about the behavior of a system during the learning process, especially while the system has not yet converged to a suitable control policy. While this may not be a critical issue in a simulated reality, it can quickly become a limiting factor when attempting to put such an algorithm in control of a system in the physical world, where certain failures, such as collisions, can result in damage that would severely hinder or even terminate the learning process, in addition to material loss or human injury. We refer to systems in which certain failure states are unacceptable as *safety-critical*.

In the last decade, learning-based control schemes have been successfully demonstrated



(a) With online guarantee validation (b) Without online guarantee validation

Figure 6.1: Hummingbird quadrotor learning a vertical flight policy under the requirement of not colliding. When the fan is turned on, the system experiences an unmodeled disturbance that it has not previously encountered. This can lead to a ground collision even under robust safety policies (right). The proposed Bayesian validation method detects the inconsistency and prevents the vehicle from entering the region of the state space where the robust model is inaccurate (left). Video: `https://youtu.be/WAAxyeSk2bw`

in robotics applications in which the safety-critical aspects were effectively removed or mitigated, typically by providing a manual fallback mechanism or retrofitting the environment to allow safe failure. In [150, 151] a trained pilot was able to remotely take over control of the autonomous helicopter at any time; the power slide car maneuvers in [152] were performed on an empty test track; and the aerobatic quadrotor in [153] was enclosed in a safety net. While mostly effective, these *ad hoc* methods tend to come with their own issues (pilot handoffs, for instance, are notoriously prone to result in accidents [154]) and do not generalize well beyond the context of the particular demonstration. It therefore seems imperative to develop principled and provably correct approaches to safety, attuned to the exploration-intense needs of learning-based algorithms, that can be built into the autonomous operation of learning robotic systems.

Current efforts in policy transfer learning propose training an initial control policy in simulation and then carrying it over to the physical system [155]. While progress in this direction is likely to reduce overall training time, it does not eliminate the risk of catastrophic system misbehavior. State-of-the-art neural network policies have been shown to be vulnerable to small changes between training and testing conditions [156], which inevitably arise

between simulated and real systems. Guaranteeing correct behavior of simulation-trained schemes in the real world thus remains an important unsolved problem.

Providing guarantees about a system's evolution inevitably requires some form of knowledge about the causal mechanisms that govern it. Fortunately, in practice it is never the case that the designer of a robotic system has no knowledge whatsoever of its dynamics: making use of approximate knowledge is both possible and, we argue, advantageous for safety. Yet, perfect knowledge of the dynamics can hardly if ever be safely assumed either. This motivates searching for points of rapprochement between data-driven and model-based techniques.

We identify three key properties that we believe any general safe learning framework should satisfy:

- ***High confidence***. The framework should be able to keep the system safe with high probability given the available knowledge about the system and the environment.
- ***Modularity***. The framework should work in conjunction with an arbitrary learning-based control algorithm, without requiring modifications to said algorithm.
- ***Minimal intervention***. The framework should not interfere with the learning process unless deemed strictly necessary to ensure safety, and should return control to the learning algorithm as soon as possible.

We can use these criteria to evaluate the strengths and shortcomings of existing approaches to safety in intelligent systems, and place our work in the context of prior research.

Related Work

Early proposals of safe learning date back to the turn of the century. Lyapunov-based reinforcement learning [157] allowed a learning agent to switch between a number of precomputed "base-level" controllers with desirable safety and performance properties; this enabled solid theoretical guarantees at the expense of substantially constraining the agent's behavior; in a similar spirit, later work has considered constraining policy search to the space of stabilizing controllers [158].

In risk-sensitive reinforcement learning [159], the expected return was heuristically weighted with the probability (risk) of reaching an "error state"; while this allowed for more general learning strategies, no guarantees could be derived from the heuristic effort. Nonetheless, the ideal problem formulation proposed in the paper, to maximize performance subject to some maximum allowable risk, inspired later work (see [160] for a survey) and is very much aligned with our own goals.

More recently, [137] proposed an ergodicity-based safe exploration policy for Markov decision processes (MDPs) with uncertain transition measures, which imposed a constraint on the probability, under the current belief, of being able to return to the starting state. While practical online methods for updating the system's belief on the transition dynamics

are not discussed, and the toy grid-world demonstrations fall short of capturing the criticality of dynamics in many real-world safety problems, the probabilistic safety analysis is extremely powerful, and our work certainly takes inspiration from it. Recent safe exploration efforts in robotics use Gaussian processes to model uncertain dynamics, but restrict safety analysis to local stability (i.e. region of attraction) and do not consider state constraints [139, 161].

The learning-based model-predictive control approach proposed in [162] adaptively learns a model of the system dynamics to improve the performance of a receding-horizon control scheme, and uses a fixed *a priori* linear model to robustly enforce state constraints. The technique was successfully demonstrated on problems with nonlinear dynamics, including quadrotor flight. The work presented here can be seen as a generalization of these ideas to nonlinear safety analysis and arbitrary learning-based control schemes.

As we saw in Chapter 2 and have been exploring throughout Part I, one powerful approach to explicitly account for model uncertainty is to cast the safety problem as a differential game [49], in which the controller must keep the system within the specified state constraints in spite of the actions of an adversarial disturbance. The optimal solution to this reachability game can then be obtained through Hamilton-Jacobi methods [52, 53] and used to robustly enforce safety, as long as the dynamical model approximates the evolution of the physical system with some bounded error. In the interior of the computed *safe set*, then, the controller can execute any desired action, as long as the safe control is applied as the system reaches the set's boundary. Similar controlled invariance properties can also be obtained through alternative methods such as "barrier functions" [163–165]. The particular advantage of the Hamilton-Jacobi game-theoretic approach is that its associated safe set is maximal (it contains *all* states that can be robustly kept safe), and therefore enforcing the safety-preserving control action at the safe set boundary constitutes a *least-restrictive* control law, that is, one that only restricts the allowable control inputs at states where this is required to robustly guarantee safety given the model uncertainty. This scheme therefore naturally lends itself to minimally constrained learning-based control. Initial work exploring this approach was presented in [166, 167].

The above methods are subject to the fundamental limitation of any model-based safety analysis, namely, the contingency of guarantees on the validity of the model. This forces designers to make early assumptions about the future operating conditions of the system, facing them with a difficult tradeoff. On the one hand, if they assume overly conservative bounds on model error, this will reduce the computed safe set and unnecessarily hinder the progress of the learning algorithm. If, on the other hand, the assumed bounds fail to capture the evolution of the physical state, the theoretical guarantees derived from the model may no longer apply to the system's operation.

Contribution

In this chapter we introduce a general safety framework that combines model-based control-theoretical analysis with data-driven Bayesian inference to construct and maintain high-probability guarantees around an arbitrary learning-based control algorithm. Drawing on

Hamilton-Jacobi robust optimal control techniques, it defines a *least-restrictive* supervisory control law, which allows the system to freely execute its learning-based policy almost everywhere, but imposes a computed action at states where it is deemed critical for safety. The safety analysis is refined through Bayesian inference in light of newly gathered evidence, both avoiding excessive conservativeness and improving reliability by rapidly imposing the computed safe actions if confidence in model-based guarantees decreases due to unexpected observations. To our knowledge this is the first work in the area of reachability analysis that reasons online about the validity of computed guarantees and uses a resilient mechanism to continue exploiting them under inaccurate prior assumptions on model error.

Our framework relies on reachability analysis for the model-based safety guarantees, and on Gaussian processes for the online Bayesian analysis. It is important to acknowledge that both of these techniques are computationally intensive and scale poorly with the dimensionality of the underlying continuous spaces, which can generally limit their applicability to complex dynamical systems. However, recent compositional approaches have dramatically increased the tractability of lightly coupled high-dimensional systems [14, 56, 168, 169], while new analytic solutions entirely overcome the "curse of dimensionality" in some relevant cases [95, 170]. The key contribution of this work is in the principled methodology for incorporating safety into learning-based systems: we thus focus our examples on problems of low dimensionality, implicitly bypassing the computational issues, and note that our method can readily be used in conjunction with these decomposition techniques to extend its application to more complex systems.

We demonstrate our method on a quadrotor vehicle learning to track a vertical trajectory close to the ground (Figure 6.1), using a policy gradient algorithm [171]. The reliability of our method is evidenced under uninformative policy initializations, inaccurate safe set estimation and strong unmodeled disturbances.

The chapter is organized as follows: In Section 6.1 we introduce the modeling framework and formally state the safe learning problem. Section 6.2 discusses the differential game analysis and derives some important invariance properties. The proposed methodology is described in Section 6.3 with the proofs of its fundamental guarantees, as well as a computationally tractable alternative with weaker but practically useful properties. Lastly, in Section 6.4 we present the experimental results.

6.1 Problem Formulation

6.1.1 System Model and State-Dependent Uncertainty

This chapter combines the robust safety analysis followed throughout Part I with Bayesian analysis to reason about the reliability of the robust guarantees in view of the observed system behavior. From the robust standpoint, we will treat the evolution of the state non-deterministically and carry out the usual worst-case analysis; from the Bayesian perspective,

we will assume that the underlying dynamics are in fact deterministic but *unknown* to the controller, and thus use probabilistic calculus to reason about the associated uncertainty.

We can formalize this with our usual dynamical system with control and disturbance inputs.

$$\dot{x} = f(x, u, d) \ . \tag{6.1}$$

In this context, d is thought of as a deterministic state-dependentdisturbance capturing unmodeled dynamics, given by an unknown Lipschitz function $d : \mathbb{R}^n \to \mathcal{D}$. The flow field $f : \mathbb{R}^n \times \mathcal{U} \times \mathcal{D} \to \mathbb{R}^n$ is assumed uniformly continuous and bounded, as well as Lipschitz in x and d for all u: this ensures that the unknown underlying dynamics $f^*(x, u) := f\big(x, u, d(x)\big)$ are Lipschitz in x. With this, state trajectories under the unknown dynamics f^* are always well defined (in the Carathéodory sense) for any measurable control input signal $\mathbf{u}$ (and as usual, in the case of f, for any measurable disturbance signal $\mathbf{d}$).

Since $d(x)$ is unknown, we attempt to bound it at each state by a compact set, allowing this bound to vary in the state space. We define the set-valued map $\hat{\mathcal{D}} : \mathbb{R}^n \rightrightarrows \mathcal{D}$, assigning a compact set $\hat{\mathcal{D}}(x) \subseteq \mathcal{D}$ to each state $x \in \mathbb{R}^n$. We will use the notation $\mathbf{x}^{\mathbf{u},\mathbf{d}}_{x,\hat{\mathcal{D}}}$ to denote the state trajectory $t \mapsto x$ corresponding to the initial condition $x \in \mathbb{R}^n$, the control signal $\mathbf{u} \in \mathbb{U}$ and the disturbance signal $\mathbf{d} \in \mathbb{D}$, subjecting the latter to satisfy $\mathbf{d}(t) \in \hat{\mathcal{D}}\big(\mathbf{x}^{\mathbf{u},\mathbf{d}}_{x,\hat{\mathcal{D}}}(t)\big)$ for all $t \geq 0$. For the interested reader, sufficient conditions on the map $\hat{\mathcal{D}}$ to ensure that the resulting Carathéodory trajectories remain well defined are discussed in the Appendix at the end of this chapter.

In Section 6.2, we present a robust, least-restrictive safety control law that enforces constraint satisfaction when the unknown underlying system dynamics satisfy $d(x) \in \hat{\mathcal{D}}(x)$ globally, and further prove stronger results in which this condition can be relaxed. In Section 6.3, we present a Bayesian approach to find a high-confidence bound $\hat{\mathcal{D}}(x)$ based on a Gaussian process model of $d(x)$. Our overall approach therefore combines robust (worst-case) guarantees with Bayesian (probabilistic) analysis, by constructing the disturbance bound to reflect the local uncertainty around the inferred disturbance function.[1]

Any model-based safety guarantees for the system will require that the bound $\hat{\mathcal{D}}$ correctly captures the unknown part of the dynamics given by $d(x)$, at least at some critical set of states x (discussed in Section 6.2). One key insight in this work is that the system should take action to ensure safety not only when the model predicts that this action may be necessary, but also when the system detects that the model itself may become unreliable in the near future.

We state here a preliminary result that will be useful later on in the chapter, and introduce the notion of *local model reliability*.

[1] Alternative methods to providing the disturbance bound $\hat{\mathcal{D}}$ are possible (for example, a conservative *a priori* estimate, or a different system identification procedure), provided they satisfy the sufficient trajectory existence conditions in the Appendix.

Proposition 6.1. *If $d(x) \in \operatorname{int} \hat{\mathcal{D}}(x)$ and the set-valued map $\hat{\mathcal{D}} : \mathbb{R}^n \to 2^{\mathcal{D}}$ is Lipschitz-continuous under the Hausdorff metric[2], then there exists $\delta > 0$ such that all possible trajectories followed by the system starting at x will satisfy $d\big(\mathbf{x}(\tau)\big) \in \hat{\mathcal{D}}\big(\mathbf{x}(\tau)\big)$ for all $\tau \in [t, t+\delta]$.*

Proof. Let $L_{\hat{\mathcal{D}}}$ be the Lipschitz (Hausdorff) constant of $\hat{\mathcal{D}}$, L_d the Lipschitz constant of d, and C_f a norm bound on the dynamics f. We then have that over an arbitrary time interval $[t, t+\delta]$, regardless of the control and disturbance signals $\mathbf{u}$, $\mathbf{d}$, any system trajectory starting at $\mathbf{x}(t) = x$ satisfies $|\mathbf{x}(\tau) - x| \leq C_f\delta, \forall \tau \in [t, t+\delta]$. This implies both $|d(\mathbf{x}(\tau)) - d(x)| \leq L_d C_f \delta$ and $d_H\big(\hat{\mathcal{D}}(\mathbf{x}(\tau)), \hat{\mathcal{D}}(x)\big) \leq L_{\hat{\mathcal{D}}} C_f \delta$. Requiring that the open ball $B\big(d(x), (L_d + L_{\hat{\mathcal{D}}})C_f\delta\big)$ be contained in $\hat{\mathcal{D}}(x)$ ensures $d(\mathbf{x}(\tau)) \in \hat{\mathcal{D}}(\mathbf{x}(\tau))$. Since $d(x) \in \operatorname{int} \hat{\mathcal{D}}(x)$, there must exist a small enough $\delta > 0$ for which this condition is met. □

We can further quantify this δ through the signed distanceto $\hat{\mathcal{D}}(x)$ at the current $d(x)$, denoted $s_{\hat{\mathcal{D}}(x)}\big(d(x)\big)$.

Corollary 6.1. *If the Lipschitz constants are known, then $d(x) \in \operatorname{int} \hat{\mathcal{D}}(x)$ implies $d(\mathbf{x}(\tau)) \in \hat{\mathcal{D}}(\mathbf{x}(\tau))$ for all times $\tau \in [t, t+\delta]$, with*

$$\delta = \frac{-s_{\hat{\mathcal{D}}(x)}\big(d(x)\big)}{(L_d + L_{\hat{\mathcal{D}}})C_f} \ .$$

The disturbance bounds $\hat{\mathcal{D}}$ derived in this chapter satisfy the hypothesis of Proposition 6.1 (see Appendix for details), and we thus refer to the condition $d(x) \in \operatorname{int} \hat{\mathcal{D}}(x)$ as the model being *locally reliable* at x.

For the remainder of the chapter, we will assume that the effect of the disturbance on the dynamics is independent of the action applied by the controller.

$$\dot{x} = f\big(x, u, d(x)\big) = f_x\big(x, u\big) + f_d\big(d(x)\big) \ . \tag{6.2}$$

with $f_x : \mathbb{R}^n \times \mathcal{U} \to \mathbb{R}^n, f_d : \mathcal{D} \to \mathbb{R}^n$, where f_x, and f_d inherit Lipschitz continuity in their first argument from f and f_d is injective onto its image. This decoupling assumption, made for ease of exposition, is not strictly necessary, and the theoretical results in this chapter can be easily adapted to the coupled case.

6.1.2 State Constraints

A central element in our problem is the *constraint set*, which defines a region $\mathcal{K} \subseteq \mathbb{R}^n$ of the state space where the system is required to remain throughout the learning process. This set is assumed closed and time-invariant; no further assumptions (boundedness, connectedness, convexity, etc.) are made. As in previous chapters, we can equivalently refer to the complement of the constraint set as the *failure set* $\mathcal{F} = \mathcal{K}^{\mathsf{c}}$.

[2]The Hausdorff metric (or Hausdorff distance) between any two sets A and B in a metric space (M, d_M) is defined as $d_H(A, B) = \max\{ \sup_{a \in A} \inf_{b \in B} d_M(a, b), \sup_{b \in B} \inf_{a \in A} d_M(a, b) \}$.

Following Chapter 2, Section 2.3, we can encode the safety *game of kind* through an auxiliary *game of degree* by implicitly characterizing $\mathcal{K}$ as the zero superlevel set of a bounded, Lipschitz *surface function* $g : \mathbb{R}^n \to \mathbb{R}$:

$$x \in \mathcal{K} \iff g(x) \geq 0 \ , \tag{6.3}$$

and using the functional

$$\mathcal{V}^{\mathbf{u},\mathbf{d}}_{\hat{\mathcal{D}}}(x) := \inf_{t \geq 0} g\big(\mathbf{x}^{\mathbf{u},\mathbf{d}}_{x,\hat{\mathcal{D}}}(t)\big) \tag{6.4}$$

to encode whether a given trajectory *ever* violates the constraints. Indeed, by Proposition 2.4, the set of states from which the trajectory stays clear of the failure set $\mathcal{F}$ under input signals $\mathbf{u}, \mathbf{d}$ is exactly the zero superlevel set of $\mathcal{V}^{\mathbf{u},\mathbf{d}}_{\hat{\mathcal{D}}}$.

6.1.3 Objective: Safe Learning

Learning-based control aims to achieve desirable system behavior by autonomously improving a control policy $\pi_g : \mathbb{R}^n \to \mathcal{U}$, typically seeking to optimize an objective function. Safe learning additionally requires that certain constraints $\mathcal{K}$ remain satisfied while searching for such a policy. With full knowledge of the system dynamics $f^*(x, u) = f\big(x, u, d(x)\big)$, we would like to find a safe control policy $\pi^* : \mathbb{R}^n \to \mathcal{U}$ producing trajectories $\mathbf{x}(t) \in \mathcal{K}$, $\forall t \geq 0$, for the largest set of initial states $x = \mathbf{x}(0)$, then restrict any learned policy so that $\pi_g(x) = \pi^*(x)$ wherever required to ensure safety. When $d(x)$ is not known exactly, however, this version of the problem cannot be solved.

Instead, given an estimated disturbance set $\hat{\mathcal{D}}(x)$ that bounds our modeling error with high confidence, we can find an inner approximation of the set of safe states by considering all the possible trajectories that can be produced under the bounded uncertainty $d(x) \in \hat{\mathcal{D}}(x)$. Our goal, then, is to find the set of *robustly safe* states x for which there exists a control policy π^* that can keep the closed-loop system evolution in $\mathcal{K}$, and consistently limit π_g to ensure that π^* is applied when necessary.

To formally state this, we introduce an important notion from robust control theory using the machinery from differential games established in Chapter 2.

Definition 6.1. *A subset $\mathcal{M} \subset \mathbb{R}^n$ is a* robust controlled invariant set *under uncertain dynamics $\dot{x} = f(x, u, d)$, $d \in \hat{\mathcal{D}}(x)$, if the controller can keep all trajectories originating in $\mathcal{M}$ from ever leaving $\mathcal{M}$ regardless of the realization of the disturbance. That is, from any state $x \in \mathcal{M}$ and for any non-anticipative disturbance strategy $\boldsymbol{\delta} \in \mathfrak{D}$, there is a control signal $\mathbf{u}$ such that for all future t,$\mathbf{x}^{\mathbf{u},\boldsymbol{\delta}[\mathbf{u}]}_{x,\hat{\mathcal{D}}}(t) \in \mathcal{M}$.*

As discussed in Chapter 2 and further formalized Chapter 3, Hamilton-Jacobi analysis allows us to construct a feedback policy $\pi^* : \mathbb{R}^n \to \mathcal{U}$ that, when implemented as a digital (sampled-data) controller, arbitrarily approaches the game-theoretic outcome for a sufficiently fast control cycle while always leading to well-defined Carathéodory trajectories.[3]

[3] A more detailed analysis for sampled-data systems can be found in [46, 47].

We will therefore discuss the design of our safe learning scheme in terms of feedback policies while keeping in mind that the underlying mathematical results rest on the game-theoretic analysis with measurable control signals and non-anticipative disturbance strategies.

Given that trajectories are continuous, the system state can only leave $\mathcal{M}$ by crossing its boundary $\partial\mathcal{M}$. Hence if $\mathcal{M}$ is closed, applying the safety-preserving feedback policy $\pi(x)$ for $x \in \partial\mathcal{M}$ is enough to render $\mathcal{M}$ robust controlled invariant, allowing an arbitrary control action to be applied in the interior of $\mathcal{M}$.

Definition 6.2. *The* safe set $\Omega_{\hat{\mathcal{D}}}$ *is the maximal robust controlled invariant set under uncertain dynamics* $\dot{x} = f(x, u, d)$, $d \in \hat{\mathcal{D}}(x)$, *that is contained in the constraint set* $\mathcal{K}$.

Success in safe learning therefore seems closely linked to model uncertainty: a tighter bound $\hat{\mathcal{D}}(x)$ on $d(x)$ yields a less conservative safe set $\Omega_{\hat{\mathcal{D}}}$, which in turn reduces the restrictions on the learning process. However, an estimated bound that fails to fully capture $d(x)$ may allow the system to execute control actions resulting in a constraint violation. The disturbance bound should thus be as tight as possible, to allow the system greater freedom in learning, yet wide enough to confidently capture the unknown dynamics, in order to ensure safety.

In the following two sections, we formalize this tradeoff and propose a framework to reason about safety guarantees under uncertainty. Section 6.2 poses the safety problem as a differential game between the controller and an adversarial disturbance, presenting a stronger result than commonly used in the reachability safety literature, which exploits the entire value function of the game rather than only its zero level set. Section 6.3 leverages this result to provide a principled approach to global safety under model uncertainty, as well as a fast local alternative that may often be useful in practice.

6.2 Safety Analysis with Imperfect Model Error Bounds

As we saw in Chapter 2, Section 2.3, the safety problem can be posed as a two-player zero-sum differential game between the system controller and the disturbance. The safety value function

$$V_{\hat{\mathcal{D}}}(x) := \inf_{\boldsymbol{\delta} \in \mathfrak{D}} \sup_{\mathbf{u} \in \mathbb{U}} \mathcal{V}_{\hat{\mathcal{D}}}^{\mathbf{u}, \boldsymbol{\delta}[\mathbf{u}]}(x) \tag{6.5}$$

can be obtained through the viscosity solution to (2.44), appropriately modified to incorporate the state-dependent disturbance bound:

$$0 = \min \left\{ g(x) - V^-(x,t), \partial_t V^-(x,t) + \max_{u \in \mathcal{U}} \min_{d \in \hat{\mathcal{D}}(x)} \nabla_x V^-(x,t) f(x,u,d) \right\} \tag{6.6a}$$

$$V^-(x,T) = g(x) \ . \tag{6.6b}$$

The viscosity solution to this Hamilton-Jacobi variational inequality is well-defined as long as some technical conditions on the bound $\hat{\mathcal{D}}$ (specified in the Appendix) hold. Since the safety margin function g is bounded, the time-dependent value function converges as $t \to -\infty$ and our sought infinite-horizon safety value function can be obtained as

$$V_{\hat{\mathcal{D}}} = \lim_{t \to -\infty} V^{-}(x,t) \ . \tag{6.7}$$

The Hamilton-Jacobi computation induces an optimal safety policy π^* that can be used in a *least-restrictive* supervisory control framework. In particular, we can apply an arbitrary learning-based control policy $\pi_g(x)$ (which may be repeatedly updated by the corresponding learning algorithm) throughout the interior of the safe set $\Omega_{\hat{\mathcal{D}}} = \{x : V_{\hat{\mathcal{D}}}(x) \geq 0\}$, transitioning to the optimally safe control as the system approaches the boundary, as in (2.45).

The worst-case analysis effectively requires the controller to enforce the safety condition for *all* possible realizations of the disturbance input, implicitly protecting the system against all "suboptimal" disturbances as well.

6.2.1 Invariance Properties of Level Sets

Traditionally, the implicit hypothesis made to guarantee safety using a least-restrictive law in the form of (2.45) has been correctness of the estimated disturbance bound $\hat{\mathcal{D}}$ everywhere in the state space, (i.e. $d(x) \in \hat{\mathcal{D}}(x) \ \forall x \in \mathbb{R}^n$), or at least everywhere in the constraint set $\mathcal{K}$ [49], [167]. We will now argue that the necessary hypothesis for safety is in fact much less stringent, by proving an important result that we will use in the following section to strengthen the proposed safety framework and retain safety guarantees under partially incorrect models.

We begin by establishing that invariance can be robustly enforced by the controller not only on the zero superlevel set of the safety value function, but on any nonnegative superlevel set.

Proposition 6.2. *Any nonnegative superlevel set of $V_{\hat{\mathcal{D}}}(x)$ is a robust controlled invariant set with respect to $\hat{\mathcal{D}}$.*

Proof. This result is analogous to the ones we used in Chapters 4 and 5 to find a nonempty tracking error bound $\Omega^{\mathcal{E}}$. We can prove this by contradiction by initially assuming that there is some $\alpha \geq 0$ such that the associated superlevel set $\{x \in \mathbb{R}^n : V_{\hat{\mathcal{D}}}(x) \geq \alpha\}$ is not a robust controlled invariant set. From Definition 6.1, there must then exist some state x, $V_{\hat{\mathcal{D}}}(x) \geq \alpha$, in this set such that for some non-anticipative strategy $\boldsymbol{\delta}$ there is no control signal $\mathbf{u}$ that can prevent the state trajectory $\mathbf{x}_{x,\hat{\mathcal{D}}}^{\mathbf{u},\boldsymbol{\delta}[\mathbf{u}]}$ from eventually leaving the set. This means that for each possible $\mathbf{u}$, there is a future time τ at which $V_{\hat{\mathcal{D}}}\big(\mathbf{x}_{x,\hat{\mathcal{D}}}^{\mathbf{u},\boldsymbol{\delta}[\mathbf{u}]}(\tau)\big) < \alpha$. From this new state, by (6.5), there must exist a non-anticipative strategy $\boldsymbol{\delta}'$ such that, regardless of the subsequent inputs of $\mathbf{u}$ the trajectory eventually reaches a state x' with $g(x') < \alpha$. We can then construct a new non-anticipative strategy $\boldsymbol{\delta}''$ that, maps each possible $\mathbf{u}$ to the original

$\boldsymbol{\delta}[\mathbf{u}]$ for all $t < \tau[\mathbf{u}]$ and subsequently applies the corresponding $\boldsymbol{\delta}'[\mathbf{u}]$ (which as we have seen exists for every $\mathbf{u}$). Under $\boldsymbol{\delta}''$, then, there is no control signal $\mathbf{u}$ that can prevent the system trajectory from eventually reaching a state from which the safety margin g is strictly less than α. However, since $V_{\hat{\mathcal{D}}}(x) \geq \alpha$, from (6.4) and (6.5) it must be that for all $\boldsymbol{\delta} \in \mathfrak{D}$ there must be some $\mathbf{u} \in \mathbb{U}$ for which the state trajectory never reaches a state with safety margin less than $V_{\hat{\mathcal{D}}}(x) \geq \alpha$, which is a contradiction. □

This result will enable us to establish a safety scheme that relies on a family of enforceable layers of safety, namely each level set of $V_{\hat{\mathcal{D}}}$, rather than a single last-resort layer at the boundary of the safe set $\partial\Omega_{\hat{\mathcal{D}}}$.

We can further prove a stronger result that makes it possible to ensure the invariance of any superlevel set of $V_{\hat{\mathcal{D}}}$ provided that the disturbance bound $\hat{\mathcal{D}}$ is locally reliable throughout its boundary, even if there are other states at which $d(x) \notin \hat{\mathcal{D}}(x)$.

Proposition 6.3. *Let $\mathcal{Q}_\alpha := \{x \in \mathbb{R}^n : V_{\hat{\mathcal{D}}}(x) = \alpha\}$ with $\alpha \geq 0$ be any nonnegative level set of the safety function $V_{\hat{\mathcal{D}}}$, computed for some robust disturbance bound $\hat{\mathcal{D}} : \mathbb{R}^n \rightrightarrows \mathcal{D}$. Suppose that, under this disturbance bound, the feedback control policy $\pi^*_{\hat{\mathcal{D}}}$ robustly keeps the system trajectory $\mathbf{x}^{\pi^*_{\hat{\mathcal{D}}},\mathbf{d}}_{x,\hat{\mathcal{D}}(x)}$ from all initial states $x \in \mathcal{Q}_\alpha$. If $d(x) \in \operatorname{int}\hat{\mathcal{D}}(x)$, $\forall x \in \mathcal{Q}_\alpha$, then the superlevel set $\{x \in \mathbb{R}^n : V_{\hat{\mathcal{D}}}(x) \geq \alpha\}$ is rendered invariant, under the underlying dynamics $f^*(x) = f\big(x, u, d(x)\big)$, by the control policy $\pi^*_{\hat{\mathcal{D}}}$.*

Proof. Since Carathéodory trajectories are continuous, any trajectory leaving the α-superlevel set $\{x \in \mathbb{R}^n : V_{\hat{\mathcal{D}}}(x) \geq \alpha\}$ must traverse its boundary $\mathcal{Q}_\alpha$. However, for all $x \in \mathcal{Q}_\alpha$, $d(x) \in \operatorname{int}\hat{\mathcal{D}}(x)$ and thus by Proposition 6.1 this means that there exists some time interval $[t, t+\delta]$ (with $\delta > 0$) during which the trajectory $\mathbf{x}$ resulting from applying $\pi^*_{\hat{\mathcal{D}}}$ under the underlying dynamics f^* satisfies $d\big(\mathbf{x}(\tau)\big) \in \hat{\mathcal{D}}\big(\mathbf{x}(\tau)\big)$ and therefore by hypothesis, $\pi^*_{\hat{\mathcal{D}}}$ keeps $\mathbf{x}(\tau)$ in the α-superlevel set throughout this non-degenerate time interval. Since this δ will continue to apply any time the system trajectory reaches a state in $\mathcal{Q}_\alpha$, it is impossible for the state trajectory to ever leave the α-superlevel set of $V_{\hat{\mathcal{D}}}$ under control policy $\pi^*_{\hat{\mathcal{D}}}$. □

This proposition, which follows from Propositions 6.2 and 2.3 by considering the singleton $\{d(x)\}$, is an important result that will be at the core of our data-driven safety enhancement. It provides a sufficient condition for safety, but unlike the standard HJI solution, it does not readily prescribe a least-restrictive control law to exploit it: how should one determine what candidate $\alpha \geq 0$ to choose, or whether a valid $\mathcal{Q}_\alpha$ exists at all? Deciding when the safe controller should intervene and what guarantees are possible is nontrivial and requires additional analysis.

The next section proposes a Bayesian approach enabling the safety controller to reason about its confidence in the model-based guarantees described in this section. If this confidence reaches a prescribed minimum value in light of the observed data, the controller can intervene early to ensure that safety will be maintained with high probability.

6.3 Bayesian Safety Assurance

6.3.1 Safety-Learning Synergy

As we have seen, robust optimal control and dynamic game theory provide powerful analytical tools to study the safety of a dynamical model. However, it is important to realize that the applicability of any theoretically derived guarantee to the real system is contingent upon the validity of the underlying modeling assumptions; in the formulation considered here, this amounts to the state disturbance function $d(x)$ being captured by the bound $\hat{\mathcal{D}}(x)$ on at least a certain subset of the state space. The system designer therefore faces an inevitable tradeoff between risk and conservativeness, due to the impossibility of accounting for every aspect of the real system in a tractable model.

In many cases, choosing a parametric model *a priori* forces one to become overly conservative in order to ensure that the system behavior will be adequately captured: this results in a large bound $\hat{\mathcal{D}}(x)$ on the disturbance, which typically leads to a small safe set $\Omega_{\hat{\mathcal{D}}}$, limiting the learning agent's ability to explore and perform the assigned tasks. In other cases, insufficient caution in the definition of the model can lead to an estimated disturbance set $\hat{\mathcal{D}}(x)$ that fails to contain the actual model error $d(x)$, and therefore the computed safe set $\Omega_{\hat{\mathcal{D}}}$ may not in fact be controlled invariant in practice, which can end all safety guarantees.

In order to avoid excessive conservativeness and keep theoretical guarantees valid, it is imperative to have both a principled method to refine the system model based on acquired measurements and a reliable mechanism to detect and react to model discrepancies with the real system's behavior; both of these elements are necessarily data-driven. We thus arrive at what is perhaps the most important insight in this work: *the relation between safety and learning is reciprocal*. Not only is safety a key requirement for learning in autonomous systems: learning about the real system's behavior is itself indispensable to provide practical safety guarantees.

In the remainder of this section we propose a method for reasoning about the uncertain system dynamics, using Gaussian processes to regularly update the model used for safety analysis, and introduce a Bayesian approach for online validation of model-based guarantees *in between* updates. We then define an adaptive safety control strategy based on this real-time validation, which leverages the theoretical results from Hamilton-Jacobi analysis to provide stronger guarantees for safe learning under possible model inaccuracies.

6.3.2 Gaussian Process

To estimate the disturbance function $d(x)$ over the state space, we model it as being drawn from a Gaussian process. Gaussian processes are a powerful abstraction that extends multivariate Gaussian inference to the infinite-dimensional space of functions, allowing Bayesian inference based on (possibly noisy) observations of a function's value at finitely many points. We give here an overview of Gaussian process inference and direct the interested reader to [172] for a more comprehensive introduction.

A Gaussian process is a random process or field defined by a mean function $\mu : \mathbb{R}^n \to \mathbb{R}$ and a positive semidefinite covariance kernel function $k : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$. We will treat each component $d^j, j \in \{1, ..., n_d\}$, of the disturbance function as an independent Gaussian process:

$$d^j(x) \sim \mathcal{GP}(\mu^j(x), k^j(x, x')) \ . \tag{6.8}$$

A defining characteristic of a Gaussian process is that the marginal probability distribution of the function value at any finite number of points is a multivariate Gaussian. This will allow us to obtain the disturbance bound $\hat{\mathcal{D}}(x)$ as a Cartesian product of confidence intervals for the components of $d(x)$ at each state x, choosing the bound to capture a desired degree of confidence.A less conservative analysis could compute $\hat{\mathcal{D}}(x)$ using a vector-valued Gaussian process model, at the expense of heavier computation.

Gaussian processes allow incorporating new observations in a nonparametric Bayesian setting. First, assume a prior Gaussian process distribution over the j-th component of $d(\cdot)$, with mean $\mu^j(\cdot)$ and covariance kernel $k^j(\cdot, \cdot)$. The class of the prior mean function and covariance kernel function is chosen to capture the characteristics of the model (linearity, periodicity, etc), and is associated to a set of hyperparameters θ_p. These are typically set to maximize the marginal likelihood of an available set of training data, or possibly to reflect some prior belief about the system.

Next, consider N measurements $D^j = [\hat{d}_1^j, \ldots, \hat{d}_N^j]$, observed with independent Gaussian noise $\epsilon_i^j \sim \mathcal{N}(0, (\sigma_n^j)^2)$ at the points $X = [x_1, \ldots, x_N]$, i.e. $\hat{d}_i^j = d^j(x_i) + \epsilon_i^j$. Combined with the prior distribution (6.8), this new evidence induces a Gaussian process posterior; in particular, the value of d^j at finitely many points X_* is distributed as a multivariate normal:

$$\mathbb{E}[d^j(X_*) \mid D^j, X] = \mu^j(X_*) + K^j(X_*, X)(K^j(X, X) + (\sigma_n^j)^2 I)^{-1}(\hat{d}^j - \mu^j(X)) \ , \tag{6.9a}$$

$$\operatorname{cov}[d^j(X_*) \mid X] = K^j(X_*, X_*) - K^j(X_*, X)(K^j(X, X) + (\sigma_n^j)^2 I)^{-1} K^j(X, X_*) \ , \tag{6.9b}$$

where $d_i^j(X) = d^j(x_i)$, $\mu_i^j(X) = \mu^j(x_i)$, and for any X, X' the matrix $K^j(X, X')$ is defined component-wise as $K_{ik}^j(X, X') = k^j(x_i, x_k')$. Note that whenever a new batch of data X is obtained the hyperparameters of the kernel function are refitted, so the variance implicitly depends on the measurements d^j. If a single query point is considered, i.e. $X_* = \{x_*\}$, the marginalized Gaussian process posterior becomes a univariate normal distribution quantifying both the expected value of the disturbance function, $\bar{d^j}(x_*)$, and the uncertainty of this estimate, $\left(\sigma^j(x_*)\right)^2$,

$$\bar{d^j}(x_*) = \mathbb{E}[d^j(x_*) \mid D^j, X] \tag{6.10a}$$

$$\left(\sigma^j(x_*)\right)^2 = \operatorname{cov}[d^j(x_*) \mid X] \ . \tag{6.10b}$$

We can use the Bayesian machinery of Gaussian process inference to compute a *likely* bound $\hat{\mathcal{D}}(x)$ on the disturbance function $d(x)$ based on the history of commanded inputs u_i and state measurements x_i, $i \in \{1, ..., N\}$. To this effect, we assume that a method for approximately measuring the state derivatives is available (e.g. by numerical differentiation),

and denote each of these measurements by $\hat{f}_i$. Based on (6.2), we can obtain measurements of $d(x_i)$ from the residuals between the observed dynamics and the model's prediction:

$$\hat{d}(x_i) = {f_d}^{-1}\left(\hat{f}_i - f_x\big(x_i, u_i\big)\right) . \tag{6.11}$$

The residuals $D = [\hat{d}(x_1), \ldots, \hat{d}(x_N)]$ are processed through (6.9) to infer the marginal distribution of $d(x_*)$ for an arbitrary point x_*, specified by the expected value $\bar{d}^j(x_*)$ and the standard deviation $\sigma^j(x_*)$ of each component of the disturbance. This distribution can be used to construct a disturbance set $\hat{\mathcal{D}}(x_*) \subseteq \mathcal{D}$ at any point x_*; in practice, this will be done at finitely many points $x_\mathbf{i}$ on a grid, and used in the numerical reachability computation to obtain the safety function and the safe control policy.

We now introduce the design parameter p as the desired marginal probability that the disturbance $d(x)$ will belong to the bound $\hat{\mathcal{D}}(x)$ at each point x; typically, p should be chosen to be close to 1. The set $\hat{\mathcal{D}}(x)$ is then chosen for each x as follows. Let $z = \sqrt{2}\,\mathrm{erf}^{-1}(p^{1/n_d})$, where $\mathrm{erf}(\cdot)$ denotes the Gauss error function; that is, define z so that the probability that a sample from a standard normal distribution $\mathcal{N}(0,1)$ lies within $[-z, z]$ is p^{1/n_d}. We construct $\hat{\mathcal{D}}(x)$ by taking a Cartesian product of confidence intervals:

$$\hat{\mathcal{D}}(x) = \prod_{j=1}^{n_d} \left[\bar{d}^j(x) - z\sigma^j(x),\, \bar{d}^j(x) + z\sigma^j(x)\right] \quad . \tag{6.12}$$

Since each component $d^j(x)$ is given by an independent Gaussian $\mathcal{N}\left(\bar{d}^j(x), \sigma^j(x)\right)$, the probability of $d(x)$ lying within the above hyperrectangle is by construction $\left(p^{1/n_d}\right)^{n_d} = p$.

Remark 6.1. *It is commonplace to use Gaussian distributions to capture beliefs on variables that are otherwise known to be bounded. While one might object that the unbounded support of* (6.10) *contradicts our problem formulation (in which the disturbance d took values from some compact set $\mathcal{D} \subset \mathbb{R}^{n_d}$), the hyperrectangle $\hat{\mathcal{D}}(x)$ in* (6.12) *is always a compact set. Note that the theoretical input set $\mathcal{D}$ is never needed in practice, so it can always be assumed to contain $\hat{\mathcal{D}}(x)$ for all x.*

Under Lipschitz continuous prior means μ^j and covariance kernels k^j, the disturbance bound (6.12) varies (Hausdorff) Lipschitz-continuously in x, satisfying the hypotheses of Proposition 6.1. This is formalized and proved in the Appendix.

The safety analysis described in Section 6.2 can be carried out by solving the Hamilton-Jacobi equation (6.6) for $\hat{\mathcal{D}}(x)$ given by (6.12), which—based on the information available at the time of computation—will be a correct disturbance bound at any single state x with probability p. As the system goes on to gather new information, however, the posterior probability for $d(x) \in \hat{\mathcal{D}}(x)$ will change at each x (and will typically no longer equal p). More generally, we have the following result.

Proposition 6.4. *Let q be the probability that $d(x) \in \hat{\mathcal{D}}(x)$ for some state x with $V_{\hat{\mathcal{D}}}(x) \geq 0$. Then with probability at least q there exists some non-degenerate time interval $[t, t+\delta]$*

(with $\delta > 0$) during which the system trajectory $\mathbf{x}$ under the underlying dynamics f^* and the optimal safety-preserving control policy $\pi^*_{\hat{\mathcal{D}}}$ maintains or increases its computed safety value, that is, $\forall \tau \in [t, t+\delta]$, $V_{\hat{\mathcal{D}}}\big(\mathbf{x}(\tau)\big) \geq V_{\hat{\mathcal{D}}}\big(\mathbf{x}(t)\big)$.

Proof. From Proposition 6.1, if $d(x) \in \operatorname{int} \hat{\mathcal{D}}(x)$ then the specified $\delta > 0$ necessarily exists (this follows directly by an analogous argument to the proof of Proposition 6.3); since the boundary of $\hat{\mathcal{D}}(x)$ has Lebesgue measure zero, the probability of $d(x) \in \operatorname{int} \hat{\mathcal{D}}(x)$ is exactly q. In addition, in the case where, with probability $1-q$, $d(x) \notin \hat{\mathcal{D}}(x)$, the probability that the postulated $\delta > 0$ exists if is non-negative, and typically strictly positive, since usually not all disturbance values $d(x) \notin \hat{\mathcal{D}}(x)$ will be unfavorable for safety. Therefore, the total probability that $\delta > 0$ exists is greater or equal to q. □

Based on this probabilistic local invariance result, we can begin to reason about high-confidence safety assurances for the underlying system dynamics in a Bayesian framework, inherited from the Gaussian process model.

6.3.3 Online Safety Guarantee Validation

In order to ensure safety under the possibility of model mismatch with the real system, it may become necessary to intervene not only on the boundary of the computed safe set, but also whenever the observed evolution of the system indicates that the model-based safety guarantees may lose validity. Indeed, failure to take a safe action in time may lead to complete loss of guarantees if the system enters a region of the state space where the model is consistently incorrect.

While the estimated bound $\hat{\mathcal{D}}$ (Section 6.3.2) and the associated safety guarantees (Section 6.2) should be recomputed as frequently as possible in light of new evidence, this process can typically take seconds or minutes, and in some cases may even require offline computation. This motivates the need to augment model-based guarantees through an online data-driven mechanism to quickly adapt to new incoming information even as new, improved guarantees are computed.

Bayesian analysis allows us to update our belief on the disturbance function as new observations are obtained. This in turn can be used to provide a probabilistic guarantee on the validity of the safety results obtained from the robust dynamical model generated from the older observations. In the remainder of this section, we will discuss how to update the belief on the disturbance function, and then provide two different theoretical criteria for safety intervention. The first criterion provides global probabilistic guarantees, but has computational challenges associated to its practical implementation. The alternative method only provides a local guarantee, but can more easily be applied in real time.

Let us denote X_{old} and $\hat{\mathbf{d}}^{\mathbf{j}}_{\text{old}}$ as the evidence used in computing the disturbance set $\hat{\mathcal{D}}(x)$, and X_{new} and $\hat{\mathbf{d}}^{\mathbf{j}}_{\text{new}}$ as the evidence acquired online after the disturbance set is computed. Conditioned on the old evidence, the function $d^j(x)$ is normally distributed with mean and variance given by (6.9) with $X = X_{\text{old}}$ and $\hat{\mathbf{d}}^{\mathbf{j}} = \hat{\mathbf{d}}^{\mathbf{j}}_{\text{old}}$, and the disturbance set is given

by (6.12). If we also condition on the new evidence and keep the hyperparameters fixed, then the mean and variance are updated by modifying (6.9) with $X = [X_{\text{old}}, X_{\text{new}}]$ and $\hat{\mathbf{d}}^{\mathbf{j}} = [\hat{\mathbf{d}}^{\mathbf{j}}_{\text{old}}, \hat{\mathbf{d}}^{\mathbf{j}}_{\text{new}}]$.

Remark 6.2. *Performing the update requires inverting $K^j([X_{old}, X_{new}], [X_{old}, X_{new}])$. This can be done efficiently employing standard techniques: since $K^j(X_{old}, X_{old})$ has already been inverted (in order to compute the disturbance bound $\hat{\mathcal{D}}$), all that is needed is inverting the Schur Complement of $K^j(X_{old}, X_{old})$ in $K^j([X_{old}, X_{new}], [X_{old}, X_{new}])$, which has the same size as $K^j(X_{new}, X_{new})$.*

Ideally we would incorporate X_{new} and $\hat{\mathbf{d}}^{\mathbf{j}}_{\text{new}}$ to relearn the Gaussian process hyperparameters as quickly as new measurements come in: otherwise new measured disturbance values $\hat{\mathbf{d}}^{\mathbf{j}}_{\text{new}}$ will only affect the posterior mean, with the variance depending exclusively on where the measurements were made (X_{new}). However, performing this update online is computationally prohibitive. Instead, we update the hyperparameters every time a new estimated bound $\hat{\mathcal{D}}$ is produced for safety analysis, keeping them fixed in between. In practice the set X_{old} will be much larger than X_{new}, so the estimated hyperparameters would not be expected to change significantly.

Remark 6.3. *In settings where conditions are slowly time-varying, it may be desirable to give recently observed data more weight than older observations. This can naturally be encoded by the Gaussian process by appending time as an additional dimension in X: points that are distant in time would then be more weakly correlated, analogous to space.*

Based on the new Gaussian distribution, we can reason about the *posterior* confidence in the safety guarantees produced by our original safety analysis, which relied on the *prior* Gaussian distribution resulting from measurements $\hat{\mathbf{d}}^{\mathbf{j}}_{\text{old}}$ at states X_{old}.

Global Bayesian safety analysis

The strongest result available for guaranteeing safety under the present framework is Proposition 6.3, which allows the system to exploit any superzero level set $\mathcal{Q}_\alpha$ ($\alpha \geq 0$) of the safety function $V_{\hat{\mathcal{D}}}$ throughout which the model is locally correct; all that is needed is for such a $\mathcal{Q}_\alpha$ to exist for $\alpha \in [0, V_{\hat{\mathcal{D}}}(x)]$ given the current state x.

It is possible to devise a safety policy to fully exploit the sufficient condition in Proposition 6.3 in a Bayesian setting: if the posterior probability that the corollary's hypotheses will hold drops to some arbitrary *global confidence threshold* γ_0, the safe controller can override the learning agent. With probability γ_0, the corollary will still apply, in which case the system is guaranteed to remain safe for all time; even if Proposition 6.3 does not apply at this time (which could happen with probability $1 - \gamma_0$), it is still possible that the disturbance $d(x)$ will not consistently take adversarial values that force the computed safety function $V_{\hat{\mathcal{D}}}(x)$ to decrease, in which case the system may still evolve safely. Therefore, this policy guarantees a lower bound on the probability of maintaining safety for all time.

In order to apply this safety criterion, the system needs to maintain a Bayesian posterior of the sufficient condition $d(x) \in \operatorname{int} \hat{\mathcal{D}}(x)$ in Proposition 6.3. We refer to this posterior probability as the *global safety confidence* $\gamma(x; X, D^j)$, or $\gamma(x)$ for conciseness:

$$\gamma(x; X, D^j) := P\big(\exists \alpha \in [0, V_{\hat{\mathcal{D}}}(x)], \forall x \in \mathcal{Q}_\alpha : d(x) \in \hat{\mathcal{D}}(x) \mid X, D^j\big) \ . \tag{6.13}$$

Based on this, we propose the least-restrictive control law:

$$\pi(x) = \begin{cases} \pi_g(x) & \text{if } \big(\gamma(x) > \gamma_0\big) \wedge \big(V_{\hat{\mathcal{D}}}(x) > 0\big) \ , \\ \pi^*(x) & \text{otherwise} \ , \end{cases} \tag{6.14}$$

so the system applies any action it desires if the global safety confidence is above the threshold, but applies the safe controller once this is no longer the case.

Note that if confidence in the safety guarantees is restored after applying the safety action the learning algorithm will be allowed to resume control of the system. This can happen by multiple mechanisms: moving to a region with higher $V_{\hat{\mathcal{D}}}(x)$ will tend to increase the probability that *some* lower level set may satisfy the hypotheses of Proposition 6.3; moving to a region with less inconsistency between expected and observed dynamics will typically lead to higher posterior belief that *nearby* level sets will satisfy the hypotheses of Proposition 6.3; and generally acquiring new data may, in some cases, increase the posterior confidence that Proposition 6.3 may apply.

Computing the joint probability that the bound $\hat{\mathcal{D}}(x)$ captures the Gaussian process $d(x)$ *everywhere* on a level set $\mathcal{Q}_\alpha$ is not possible, since the set of functions $d(x)$ satisfying this condition is bounded on uncountably many dimensions, and thus not measurable in function space. Similarly, evaluating the joint probability for a continuum of level sets $\mathcal{Q}_\alpha$ for $\alpha \in [0, V_{\hat{\mathcal{D}}}(x)]$ is not feasible. Instead, exploiting the Lipschitz assumption on $d(x)$, we can obtain the sought probability $\gamma(x)$ from a marginal distribution over a sufficiently dense set of sample points on each $\mathcal{Q}_\alpha$ and a sufficiently dense collection of level sets between 0 and $V_{\hat{\mathcal{D}}}(x)$.

We can then use numerical methods [173] to compute the multivariate normal cumulative distribution function and estimate the marginal probability (using compact logic notation):

$$\gamma(x) \approx P\Bigg(\bigvee_{s=1}^{S} \bigwedge_{i=1}^{I} d(x_{s,i}) \in \hat{\mathcal{D}}(x_{s,i})\Bigg) \ , \tag{6.15}$$

over S level sets $0 = \alpha_0 < \ldots < \alpha_S = V_{\hat{\mathcal{D}}}(x)$ and I sample points from each level set Q_{α_s}. As the density of samples increases with larger S and I, the marginal probability (6.15) asymptotically approaches the Gaussian process probability (6.13). Unfortunately, however, current numerical methods can only efficiently approximate these probabilities for multivariate Gaussians of about 25 dimensions [173], which drastically limits the number of sample points ($S \times I \approx 25$) that the marginal probability can be evaluated over, making it difficult to obtain a useful estimate. In view of this, a viable approach may be to bound

(6.13) below as follows:

$$\gamma(x) \geq \underline{\gamma}(x) := \max_{\alpha \in [0, V_{\hat{\mathcal{D}}}(x)]} P\big(\forall x \in \mathcal{Q}_\alpha : d(x) \in \hat{\mathcal{D}}(x)\big)\,, \tag{6.16}$$

and approximately compute this as

$$\underline{\gamma}(x) \approx \max_{s \in \{1,\ldots,S\}} P\Big(\bigwedge_{i=1}^{I} d(x_{s,i}) \in \hat{\mathcal{D}}(x_{s,i})\Big)\ , \tag{6.17}$$

with the advantage that a separate multivariate Gaussian evaluation can be done now for each level set ($I \approx 25$). Computing this approximate probability as the system explores its state space provides a decision mechanism to guarantee safe operation of the system with a desired degree of confidence, which the system designer or operator can adjust through the γ_0 parameter.

Local Bayesian safety analysis

Evaluating the expression in (6.17) is still computationally intensive, which can limit the practicality of this method for real-time validation of safety guarantees in some applications, such as mobile robots relying on on-board processing. An alternative is to replace the global safety analysis with a local criterion that offers much faster computation traded off with a weaker safety guarantee.

Instead of relying on Proposition 6.3, this lighter method exploits Propositions 6.1 and 6.4. The system is allowed to explore the computed safe set freely as long as the probability of the estimated model $\hat{\mathcal{D}}$ being *locally reliable* remains above a certain threshold λ_0; if this threshold is reached, the safe controller intervenes, and the system is guaranteed to locally maintain or increase the computed safety value $V_{\hat{\mathcal{D}}}(x)$ with probability no less than λ_0. While this local guarantee does not ensure safety globally, it does constitute a useful heuristic effort to prevent the system from entering unexplored and potentially unsafe regions of the state space. Further, although the method is not explicitly tracking the hypotheses of Proposition 6.3, the local result becomes a global guarantee if these hypotheses do indeed hold.

We define the *local safety confidence* $\lambda(x; X, D^j)$, more concisely $\lambda(x)$, as the posterior probability that $d(x)$ will be contained in $\hat{\mathcal{D}}(x)$ at the current state x, given all observations made until now:

$$\lambda(x; X, D^j) := P\big(d(x) \in \hat{\mathcal{D}}(x) \mid X, D^j\big)\ . \tag{6.18}$$

We then have the following local safety certificate.

Proposition 6.5. *Let the disturbance $d(\cdot)$ be distributed component-wise as n_d independent Gaussian processes* (6.8)*. The safety policy π^* is guaranteed to locally maintain or increase the system's computed safety $V_{\hat{\mathcal{D}}}$ with probability greater than or equal to the local safety confidence $\lambda(x)$.*

Proof. The proof follows directly from Propositions 6.1 and 6.4, and the definition of $\lambda(x)$, noting that the boundary of $\hat{\mathcal{D}}(x)$ has zero Lebesgue measure and thus under any Gaussian distribution $P(d \in \operatorname{int} \hat{\mathcal{D}}(x) \mid d \in \hat{\mathcal{D}}(x)) = 1$. □

A *local confidence threshold* $\lambda_0 \in (0, p)$ can be established such that whenever $\lambda(x) < \lambda_0$ the model is considered insufficiently reliable (reachability guarantees may fail locally with probability greater than $1 - \lambda_0$), and the safety control is applied. The proposed safety control strategy is therefore as follows:

$$\pi(x) = \begin{cases} \pi_g(x) & \text{if } \left(V_{\hat{\mathcal{D}}}(x) > 0\right) \wedge \left(\lambda(x) > \lambda_0\right) \ , \\ \pi^*(x) & \text{otherwise} \ , \end{cases} \tag{6.19}$$

Similarly to (6.14), under this control law, if confidence on the local reliability of the model is restored after applying the safe action and making new observations, the system will be allowed to resume its learning process, as long as it is in the interior of the computed safe set.

After generating a new Gaussian process model and defining $\hat{\mathcal{D}}(x)$ as described in Section 6.3.2, the prior probability with which the disturbance function $d(x)$ belongs to the set $\hat{\mathcal{D}}(x)$ is by design p everywhere in the state space. As the system evolves, more evidence is gathered in the form of measurements of the disturbance along the system trajectory, so that the belief that $d(x) \in \hat{\mathcal{D}}(x)$ is updated for each x. In particular, in the Gaussian process model, this additional evidence amounts to augmenting the covariance matrix K^j in (6.9) with additional data points and reevaluating the mean and variance of the posterior distribution of $d(x)$. Based on the new Gaussian distribution, $\lambda(x; X, D^j)$ can readily be evaluated for each x as

$$\lambda(x) = \prod_{j=1}^{n_d} \frac{1}{2} \left[\operatorname{erf}\left(\frac{d^j_+(x) - m^j(x)}{s^j(x)\sqrt{2}} \right) - \operatorname{erf}\left(\frac{d^j_-(x) - m^j(x)}{s^j(x)\sqrt{2}} \right) \right] \ , \tag{6.20}$$

with $d^j_+(x) = \bar{d}^j(x) + z\sigma^j(x)$, $d^j_-(x) = \bar{d}^j(x) - z\sigma^j(x)$, $m^j(x) = \mathbb{E}[d^j(x) \mid X, D^j]$, $s^j(x) = \sqrt{\operatorname{var}(d^j(x) \mid X)}$; recall that z was defined to yield the desired probability mass p in $\hat{\mathcal{D}}(x)$ at the time of safety computation, as per (6.12).

Parameters p and λ_0 (or, in its case, γ_0) allow the system designer to choose the degree of conservativeness in the system: while p regulates the amount of uncertainty accounted for by the robust model-based safety computation, λ_0 (γ_0) determines the acceptable degradation in the resulting certificate's posterior confidence before a safety intervention is initiated. A value of p close to 1 will lead to a large, high-confidence $\hat{\mathcal{D}}(x)$ throughout the state space, but this analysis may result in a small or even empty safe set; on the other hand, if p is low, $\hat{\mathcal{D}}(x)$ will be smaller and the computed safe set will be larger, but guarantees are more likely to be deemed unreliable (as per λ_0 or γ_0) in light of later observations.

In the case of local safety analysis, immediately after computing a new model $\hat{\mathcal{D}}$, $\lambda(x)$ is by construction equal to p everywhere in the state space. As more measurements are obtained,

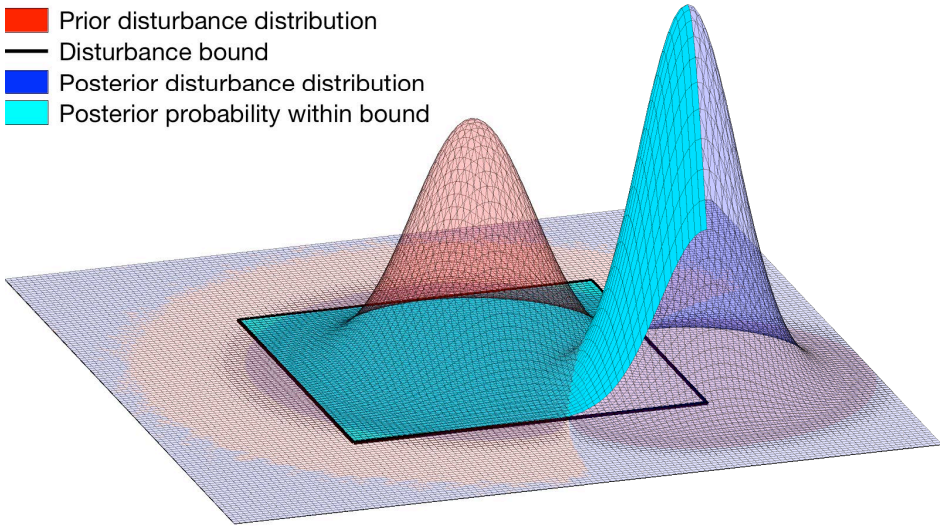


Figure 6.2: Evolution of the probability distribution of the disturbance $d(x)$ at a particular state x under Bayesian updates of the underlying Gaussian process. The prior distribution is used to compute the bound $\hat{\mathcal{D}}(x)$ using confidence intervals, such that it contains a specified probability mass p. As more data are obtained, the distribution may shift, leading to a different posterior probability mass contained within $\hat{\mathcal{D}}(x)$.

the posterior distribution over the disturbance changes, as illustrated in Figure 6.2, which can result in $\lambda(x)$ locally increasing or decreasing. If λ_0 is chosen to be close to p, it is likely that the safety override will take place under minor deviations with respect to the model's prediction; as λ_0 becomes lower, however, the probability that the disturbance will violate the modeling assumptions before the safety controller intervenes increases. This reflects the fundamental tradeoff between risk and conservativeness in safety-critical decision making under uncertainty. The proposed framework therefore allows the system designer to adjust the degree of conservativeness according to the needs and characteristics of the system at hand.

6.4 Experimental Results

We demonstrate our framework on a practical application with an autonomous quadrotor helicopter learning a flight controller in different scenarios. Our method is tested on the Stanford-Berkeley Testbed of Autonomous Rotorcraft for Multi-Agent Control (STARMAC), using Ascending Technologies Pelican and Hummingbird quadrotors (Figure 6.1). The system receives full state feedback from a VICON motion capture system. For the purpose of this series of experiments, the vehicle's dynamics are approximately decoupled through an on-board controller responsible for providing lateral stability around hover and vertical flight; our framework is then used to learn the feedback gains for a hybrid verti-

cal flight controller. The learning and safety controllers were implemented and executed in MATLAB, on a Lenovo Thinkpad with an Intel i7 processor that communicated wirelessly with the vehicle's 1.99 GHz Quadcore Intel Atom processor. This was all done using the Indigo version of the Robot Operating System (ROS) framework. Reachability computations are executed using the Level Set Toolbox [53], employing the Lax-Friedrichs approximation for the numerical Hamiltonian; a weighted essentially nonoscillatory scheme for spatial derivatives; and a third-order total variation diminishing Runge-Kutta scheme for the time derivative [52]. Once the safety function and safety policy have been computed, they are stored as look-up tables that can be quickly consulted in constant time.

The purpose of the results presented here is not to advance the state of the art of quadrotor flight control or reinforcement learning techniques, but to illustrate how the proposed method can allow safe execution of an arbitrary learning-based controller without requiring any particular convergence rate guarantees. To fully demonstrate the reliability of our safe learning framework, in our first setup we let the vehicle begin its online learning in mid-air starting with a completely untrained controller. The general functioning of the framework can be observed in the second flight experiment, in which the vehicle starts with a conservative model and iteratively computes empirical estimates of the disturbance, gradually expanding its computed safe set while avoiding overreliance on poor predictions. Finally, we include an experiment in which an unexpected disturbance is introduced into the system. The vehicle reacts by immediately applying the safe action dictated by its local safety policy and retracting from the perturbed region, successfully maintaining safety throughout its trajectory. We show how the absence of online guarantee validation in the same scenario can result in loss of safety.

We use an affine dynamical model of quadrotor vertical flight, with state equations:

$$\begin{aligned} \dot{x}_1 &= x_2 \ , \\ \dot{x}_2 &= k_g + k_0 + k_T u + d(x) \ , \end{aligned} \tag{6.21}$$

where x_1 is the vehicle's altitude, x_2 is its vertical velocity, and $u \in [0, 1]$ is the normalized motor thrust command. The gravitational acceleration is $k_g = -9.8$ m/s^2. The parameters of the affine model k_T and k_0 are determined for the Pelican and the Hummingbird vehicles through a simple on-the-ground experimental procedure—a scale is used to measure the normal force reduction for different values of u. The state constraint $\mathcal{K} = \{x : 0 \text{ m} \leq x_1 \leq 2.8 \text{ m}\}$ encodes the position of the floor and the ceiling, which must be avoided. Finally, d is an unknown, state-dependent scalar disturbance term representing unmodeled forces in the system. We learn $d(x)$ using a Gaussian process model, and generate $\hat{\mathcal{D}}(x)$ as the marginal 95% confidence interval at each x. We implement *local* Bayesian guarantee validation, conservatively approximating (6.20) by assuming

$$s^j(x) := \sqrt{\text{var}(d^j(x) \mid X)} \approx \sqrt{\text{var}(d^j(x) \mid X_{\text{old}})} \ ,$$

that is, neglecting the (favorable but often small) reduction in uncertainty due to X_{new}. This was done for ease of prototyping.

As the learning-based controller, we choose an easily implementable policy gradient reinforcement learning algorithm [171], which learns the weights for a linear mapping from state features to control commands. Following [167], we define different features for positive and negative velocities and position errors, since the (unmodeled) rotor dynamics may be different in ascending and descending flight. This can be seen as the policy gradient algorithm learning the feedback gains for a hybrid proportional-integral-derivative (PID) controller.

6.4.1 From Fall to Flight

To demonstrate the strength of Hamilton-Jacobi-based guarantees for safely performing learning-based control on a physical system, we first require a Pelican quadrotor to learn an effective vertical trajectory tracking controller with an arbitrarily poor initialization. To do this, the policy gradient algorithm is initialized with all feature weights set to 0. The pre-computed safety controller (numerically obtained using [53]) is based on a conservative uncertainty bound of ± 1.5 m/s^2 everywhere in the state space; no new bounds are learned during this experiment. The reference trajectory requires the quadrotor to aggressively alternate between hovering at two altitudes, one of them (1.5 m) near the center of the room, the other (0.1 m) close to the floor.

This first experiment illustrates the interplay between the *learning controller* and the *safety policy*. The *iterative safety re-computation* and *Bayesian guarantee validation* components of the framework are not active here. Consistently, this demonstration uses (2.45) as the least-restrictive safe policy.

The experiment, shown in Figure 6.3, is initialized with the vehicle in mid-air. Since all feature weights are initially set to zero, the vehicle's initial action is to enter free fall. However, as the quadrotor is accelerated by gravity towards the floor, the boundary of the computed safe set is reached, triggering the intervention of the safety controller, which automatically overrides the learning controller and commands the maximum available thrust to the motors ($u = 1$), causing the vehicle to decelerate and hover at a small distance from the ground. For the next few seconds, there is some chattering near the boundary of the safe set, and the policy gradient algorithm has some occasions to attempt to control the vehicle when it is momentarily pushed into the interior of the safe set. Initially it has little success, which leads the safety controller to continually intervene to prevent the quadrotor from colliding with the floor; this has the undesirable effect of slowing down the learning process, since observations under this interference are uninformative about the behavior of the vehicle when actually executing the commands produced by the learning controller (which is an "on-policy" algorithm). However, at approximately $t = 40$ s, the learning controller is able to make the vehicle ascend towards its tracking reference, retaining control of the vehicle for a longer span of time and accelerating the learning process. By $t = 60$ s, the quadrotor is approximately tracking the reference, with the safety controller only intervening during the aggressive descent phase of the repeated trajectory, to ensure (under its conservative model) that there is no risk of a ground collision. The controller continues to learn in subsequent iterations, overall improving its tracking accuracy.

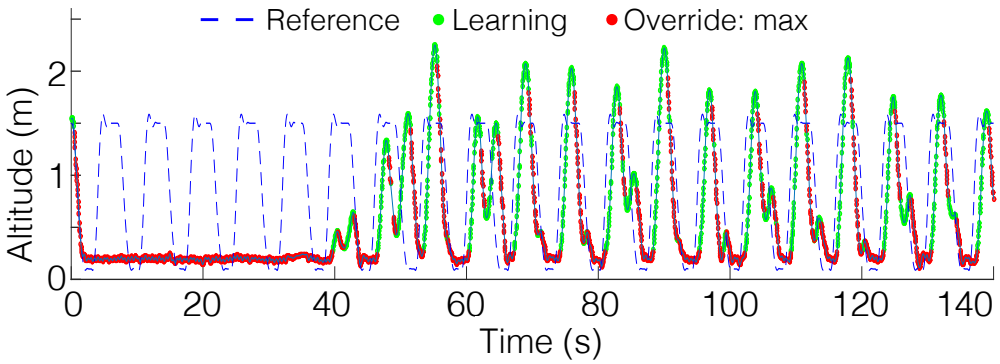


Figure 6.3: Quadrotor altitude and reference trajectory over time. Initial feedback gains are set to zero. When the learning controller (green) lets the vehicle drop, the safety control (red) takes over preventing a collision. Within a few seconds, the learned feedback gains allow rough trajectory tracking and are subsequently tuned as the vehicle attempts to minimize error.

The remarkable result in this experiment is not in the quality of the learned tracking controller after only a few seconds of active exploration (a merit that corresponds to the reinforcement learning method [171]), but the system's ability to achieve competent performance at its task from an extremely poor initial policy while remaining safe at all times.

6.4.2 When in Doubt

In the second experiment, we demonstrate the iterative updating of the safe set and safety policy using observations of the system dynamics gathered over time, as well as the online validation of the resulting guarantees. All components of the framework are active during the test, namely *learning controller*, *safety policy*, *iterative safety re-computation*, and *Bayesian guarantee validation*, with the main focus being on the latter two.

Here, the Pelican quadrotor attempts to safely track the same reference trajectory, while using the gathered information about the system's evolution to refine its notion of safety. In this case, the policy gradient learning algorithm is initialized to a hand-tuned set of parameter values. The initial dynamic model available to the safety algorithm is identical to the one used in the previous experiment, with a uniform uncertainty bound of $\pm 1.5\text{m/s}^2$. However, the system is now allowed to update this bound, throughout the state space, based on the disturbance posterior computed by a Gaussian process model.

To learn the disturbance function, the system starts with a Gaussian process prior over $d(\cdot)$ defined by a zero mean function and a squared exponential covariance function:

$$k(x, x') = \sigma_f^2 \exp\left(\frac{(x - x')^T \mathcal{L}^{-1}(x - x')}{2}\right), \tag{6.22}$$

where $\mathcal{L}$ is a diagonal matrix, with $\mathcal{L}_i$ as the ith diagonal element, and $\theta_p = \left[\sigma_f^2, \sigma_n^2, \mathcal{L}_1, \mathcal{L}_2\right]$ are the hyperparameters, σ_f^2 being the signal variance, σ_n^2 the measurement noise variance, and the $\mathcal{L}_i$ the squared exponential's characteristic length for position and velocity respectively. The hyperparameters are chosen to maximize the marginal likelihood of the training data set, and are recomputed for each new batch of data when a new disturbance model $\hat{\mathcal{D}}(x)$ is generated for safety analysis. Finally, the chosen prior mean and covariance kernel classes are both Lipschitz continuous, ensuring that all required technical conditions for the theoretical results hold (proofs are presented in the Appendix).

The expressions (6.9), (6.10) give the marginal Gaussian process posterior on $d(x^*)$ for a query point x^*. To numerically compute the safe set, the system first evaluates (6.12) to obtain the disturbance bound $\hat{\mathcal{D}}(\mathbf{x_i})$ at every point $\mathbf{x_i}$ on a state-space grid, as the 95% confidence interval ($p = 0.95$) of the Gaussian process posterior over $d(x)$; next, it performs the robust safety analysis by numerically solving the HJI equation (6.6) on this grid (using [53]) and obtaining the safety function $V_{\hat{\mathcal{D}}}(x)$.

The trajectory followed by the quadrotor in this experiment is shown in Figure 6.4. The vehicle starts off with an *a priori* conservative global bound on $d(x)$ and computes an initial conservative safe set Ω_1 (Figure 6.5). It then attempts to track the reference trajectory avoiding the unsafe regions by transitioning to the safe control $u^*(x)$ on $\partial\Omega_1$. The disturbance is measured and monitored online during this test, under the local safety confidence criterion, and found to be locally consistent with the initial conservative bound. After collecting 10 s of data, a new disturbance bound $\hat{\mathcal{D}}(x)$ is constructed using the corresponding Gaussian process posterior, from which a second safety function $V_{\hat{\mathcal{D}},2}(x)$ and safe set Ω_2 are computed via Hamilton-Jacobi reachability analysis. This process takes roughly 2 seconds, and at approximately $t = 12$ s the new safety guarantees and policy are substituted in.

The Pelican continues its flight under the results of this new safety analysis: however, shortly after, the vehicle measures values of d that consistently approach the boundary of $\hat{\mathcal{D}}(x)$, and reacts by applying the safe control policy and locally climbing the computed safety function. This confidence-based intervention takes place several times during the test run, as the vehicle measures disturbances that lower its confidence in the local model bounds, effectively preventing the vehicle from approaching the ground.

After a few seconds, a new Gaussian process posterior is computed based on the first 20 s of flight data, resulting in an estimated safe set Ω_3, an intermediate result between the initial conservative Ω_1 and the overly permissive Ω_2 (Figure 6.5). The learning algorithm is then allowed to resume tracking under this new safety analysis, and no further safety overrides take place due to loss of safety confidence.

This experiment demonstrates the algorithm's ability to safely refine its notion of safety as more data become available, without requiring the process to consist in a series of strictly conservative under-approximations.

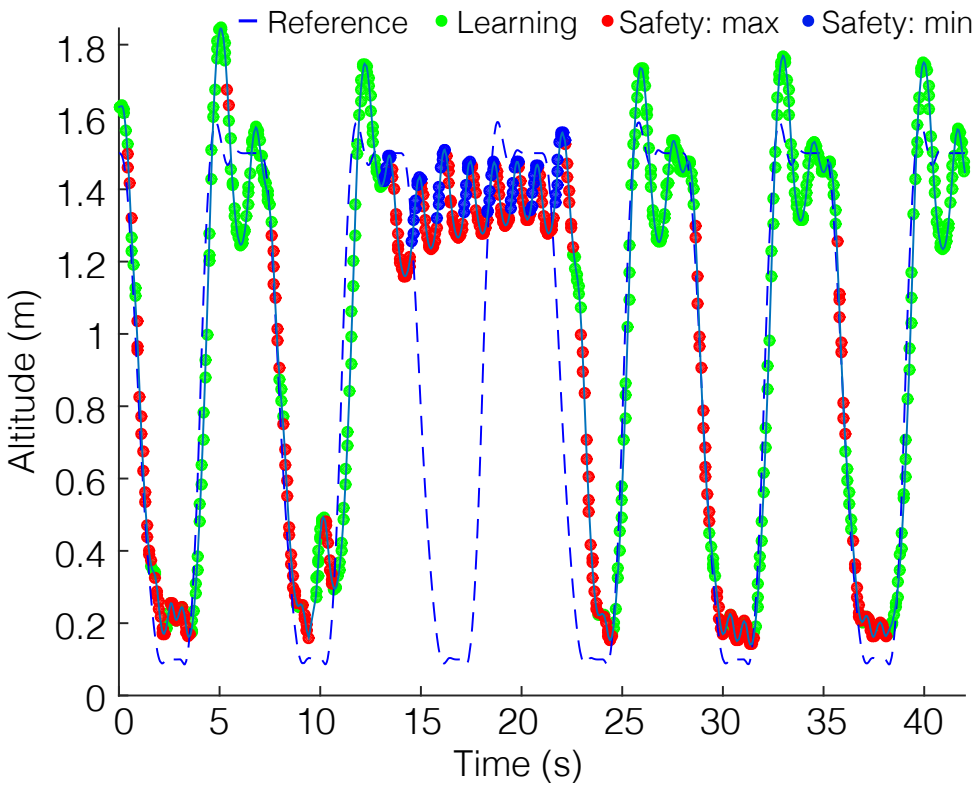


Figure 6.4: Quadrotor altitude and reference trajectory over time. After flying with an initial conservative model, the vehicle computes a first Gaussian process model of the disturbance with only a few data points, resulting in an insufficiently accurate bound. The safety policy detects the low confidence and refuses to follow the reference to low altitudes. Once a more accurate disturbance bound is computed, tracking is resumed, with a less restrictive safe set than the original one.

6.4.3 Gone with the Wind

In this last experimental result, we display the efficacy of online safety guarantee validation in handling alterations in operating conditions unforeseen by the system designer. All components of the framework are active, except for the *iterative safety re-computation*, which is not used in this case.

This experiment is performed using the lighter Hummingbird quadrotor, which is more agile than the Pelican but also more susceptible to wind. We initialize the disturbance set to a conservative range of ± 2 m/s^2, which amply captures the error in the double-integrator model for vertical flight. The vehicle tracks a slow sinusoidal trajectory using policy gradient [171] to improve upon the manually initialized controller parameters. At approximately $t = 45$ s an unmodeled disturbance is introduced by activating a fan aimed laterally at the quadrotor. The fan is positioned on the ground and angled slightly upward, so that its effect

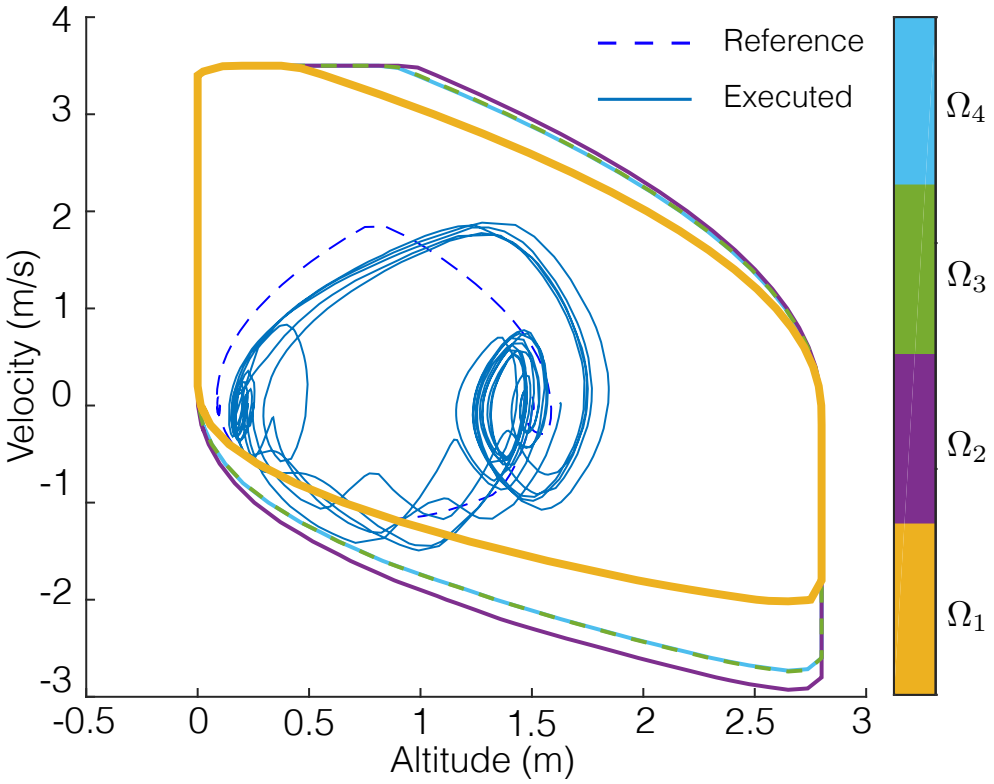


Figure 6.5: Safe sets computed online by the safety algorithm as it gathers data and successively updates its Gaussian process disturbance model. The vehicle's trajectory eventually leaves the initial, conservative Ω_1, but remains in the converged safe set (Ω_4) at all times, even *before* this set is computed. While the intermediate set Ω_2 would have been overly permissive, this is remedied by the intervention of the safety controller as soon as the model is observed to behave poorly.

increases as the quadrotor flies closer to the ground. The presence of the airflow causes the attitude and lateral position controllers to use additional control authority to stabilize the quadrotor, which couples into the vertical dynamics as an unmodeled force.

The experiment is performed with and without the *Bayesian guarantee validation* component, with resulting trajectories shown in Figure 6.6. Without validation, the quadrotor violates the constraints, repeatedly striking the ground. With validation, the fan's airflow is quickly detected as a discrepancy with the model near the floor, and the safety controller override is triggered. The vehicle avoids entering the affected region for the remainder of the flight. Although only the local confidence method is used, providing a strictly local safety guarantee, the safe controller succeeds in maintaining safety throughout the experiment. This provides strong evidence suggesting that, beyond its theoretical guarantees, the local Bayesian analysis also constitutes an effective best-effort approach to safety in more general

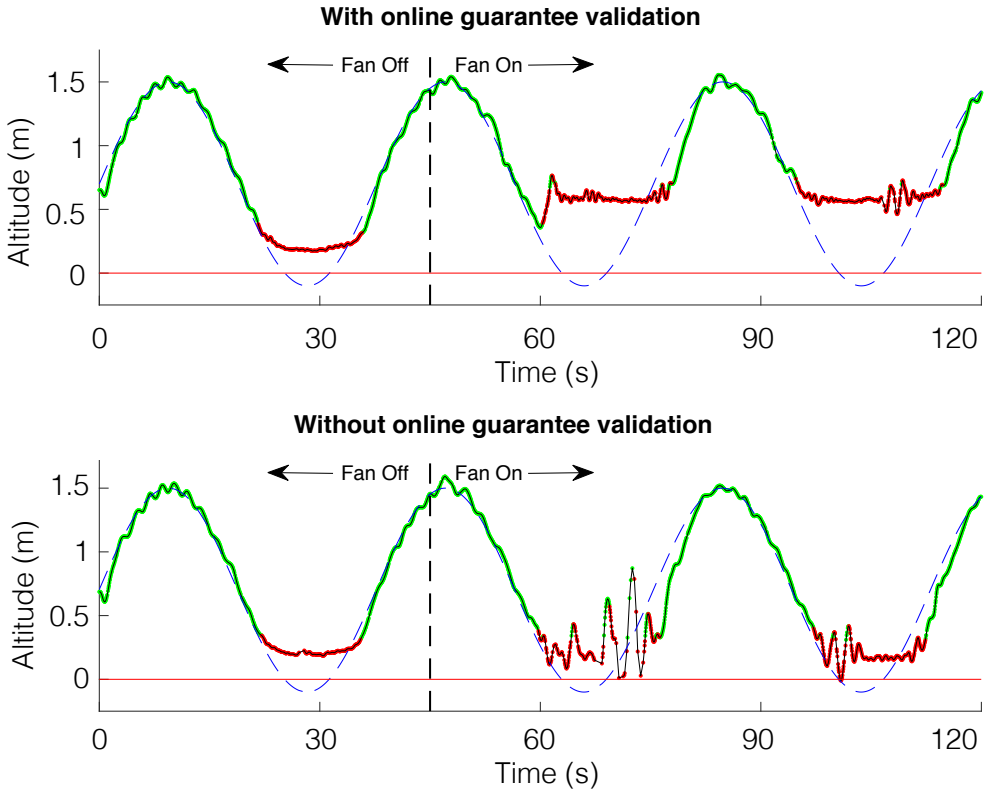


Figure 6.6: Quadrotor altitude and reference trajectory over time, shown with and without online model validation. After the fan is turned on, the vehicle checking local model reliability detects the inconsistency and overrides the learning controller, avoiding the region with unmodeled dynamics; the vehicle without model validation enters this region and collides with the ground multiple times. The behavior is repeated when the reference trajectory enters the perturbed region a second time.

conditions, given limited computational resources and available knowledge about the system.

6.5 Chapter Summary

We have introduced a safe learning framework that combines the robust reachability guarantees explored in Part I with Bayesian analysis based on empirical observations. This results in a minimally restrictive supervisory controller that can allow an arbitrary learning algorithm to safely explore its state and strategy spaces. As more data are gathered online, the framework allows the system to reason probabilistically about the validity of its robust model-based safety guarantees in light of the latest empirical evidence.

We have demonstrated the merits of this framework on a physical quadrotor system running a simple policy gradient reinforcement learning algorithm; the vehicle was able to

autonomously learn an efficient altitude controller from a poor initialization that would have otherwise led it to crash. In addition, the Gaussian process learning mechanism allowed the quadrotor to gradually refine its dynamical model over time, while the online safety validation scheme prevented overreliance on the learned model when it failed to accurately capture the observed dynamics—including a strong unmodeled disturbance introduced by turning on a fan—and enabled the quadrotor to preserve its safety assurances in spite of these inaccuracies.

A central tenet throughout Part II is that providing high-confidence safety assurances for systems that operate in uncertain environments requires a rapprochement between model-based and data-driven techniques, often regarded as a dichotomy by both theoreticians and practitioners. The work in this and the following chapters aims to provide mathematical arguments and empirical evidence of the superior potential that the two approaches hold when used in conjunction.

Scaling up safety assurances as intelligent systems achieve increasing complexity poses an important open research problem. In particular, as autonomous systems interact increasingly closely with human beings, we expect that the graceful interplay of safety and learning, combining theoretical guarantees with empirical grounding, will become central to their success. The remainder of Part II is dedicated to exploring such human-centric challenges, further developing some of the key ideas introduced in this chapter.

Appendix

Restricting one of the control inputs d to a *state-dependent* bound $\hat{\mathcal{D}}(x)$ introduces questions as to whether a unique Carathéodory solution to (6.1) continues to exist. The standard existence and uniqueness theorems [24] assume fixed control sets. If the variation in the control sets can instead be expressed through the dynamic equation itself without breaking the continuity conditions, then it is easy to extend the standard result to at least a class of space-dependent control sets. We introduce two technical assumptions, which give sufficient conditions for the existence and uniqueness of a solution to the dynamical equations, and prove that any disturbance set $\hat{\mathcal{D}}(x)$ obtained from a Gaussian process model with Lipschitz prior mean and covariance kernel satisfies these assumptions.

Assumption 6.1. *For all $x \in \mathbb{R}^n$, $\hat{\mathcal{D}}(x)$ is a closed deformation retract of $\mathcal{D}$, that is, there exists a continuous map $H_x : \mathcal{D} \times [0,1] \to \hat{\mathcal{D}}(x)$ such that for every $d \in \mathcal{D}$ and $\hat{d} \in \hat{\mathcal{D}}(x)$, $H_x(d,0) = d$, $H_x(d,1) \in \hat{\mathcal{D}}(x)$, $H_x(\hat{d},1) = \hat{d}$.*

Assumption 6.2. *Let $r : \mathbb{R}^n \times \mathcal{D} \to \mathcal{D}$ be such that $r(x,d) = H_x(d,1)$ as defined above. Then r is Lipschitz continuous in x, and uniformly continuous in d.*

Intuitively, the first assumption means that $\mathcal{D}$ can be continuously deformed into $\hat{\mathcal{D}}(x)$ for any x, while the second prevents the disturbance bound $\hat{\mathcal{D}}(x)$ from changing abruptly as one moves in the state space. The retraction map r allows us to absorb the state-dependence of the disturbance bound into the system dynamics, enabling us to use the standard analysis for differential games, which considers measurable time signals drawn from fixed input sets. This is formalized in the following result.

Proposition 6.6. *The saturated system dynamics $\tilde{f}_{\hat{\mathcal{D}}}(x,u,d) := f\big(x,u,r(x,d)\big)$ are bounded and uniformly continuous in all variables, and Lipschitz in x.*

Proof. Boundedness and uniform continuity of $\tilde{f}_{\hat{\mathcal{D}}}$ in u are trivially inherited from f; we therefore focus on d and x.

First, since r is uniformly continuous in d, and f is Lipschitz (hence uniformly continuous) in its third argument, we have by composition that $\tilde{f}_{\hat{\mathcal{D}}}$ is uniformly continuous in d.

Lipschitz continuity in x is less immediate due to its appearance in both the first and third arguments of f. Again by composition, Lipschitz continuity of r in x implies that $f\big(y,u,r(\cdot,d)\big)$ is also Lipschitz for all $y \in \mathbb{R}^n$, $u \in \mathcal{U}$ and $d \in \mathcal{D}$. Letting L_r be the Lipschitz constant of r and L_x be the Lipschitz constant of f in its first argument, we have, for any $x, \tilde{x} \in \mathbb{R}^n$:

$$
\begin{aligned}
&|f(x,u,r(x,d)) - f(\tilde{x},u,r(\tilde{x},d))| \\
&\leq |f(x,u,r(x,d)) - f(\tilde{x},u,r(x,d))| + |f(\tilde{x},u,r(x,d)) - f(\tilde{x},u,r(\tilde{x},d))| \\
&\leq (L_x + L_d L_r)|x - \tilde{x}| \ .
\end{aligned}
$$

Thus $\tilde{f}_{\hat{\mathcal{D}}}(\cdot,u,d) = f(\cdot,u,r(\cdot,d))$ is indeed Lipschitz in x. □

Corollary 6.2. *The dynamical system given by* (6.1) *with d constrained to lie in a state-dependent set $\hat{\mathcal{D}}(x)$ satisfying Assumptions 6.1 and 6.2 has a unique continuous (Carathéodory) solution.*

The above result guarantees existence and uniqueness of system trajectories for any state-dependent disturbance bound $\hat{\mathcal{D}}(\cdot)$ that satisfies Assumptions 6.1 and 6.2. Moreover, the above construction allows us to transform the system dynamics $f(x, u, d)$ with $d \in \hat{\mathcal{D}}(x)$ into the standard form with fixed input sets (i.e. $\tilde{f}_{\hat{\mathcal{D}}}(x, u, d)$ with $u \in \mathcal{U}$, $d \in \mathcal{D}$), so that all results from the differential games literature can readily be applied to our formulation.

Let us now see that the posterior mean and standard deviation of the components of $d(x)$ are Lipschitz continuous functions of the state x under our Gaussian process framework.

Proposition 6.7. *Let the prior mean function μ^j be Lipschitz continuous, and the covariance kernel k^j be jointly Lipschitz continuous, for the jth component of the disturbance function $d(x)$. Then the posterior mean $\bar{d}^j(x)$ and standard deviation $\sigma^j(x)$, as given by* (6.9), (6.10) *are Lipschitz continuous in x.*

Proof. The result follows from applying the hypotheses to (6.9), (6.10). Note that the standard deviation $\sigma^j(x)$ is the square root of the variance in (6.10); the square root function is Lipschitz everywhere except at 0, and Bayesian inference under nondegenerate prior and likelihood never results in 0 posterior variance. Thus $\sigma^j(\cdot)$ is also Lipschitz. □

The following proposition relates the state-dependent bound $\hat{\mathcal{D}}(x)$ obtained from Gaussian process regression to Assumptions 6.1 and 6.2, ensuring that the dynamical system (6.1) is well-defined, and therefore the associated dynamic game be solved using the methods presented in Section 6.2.

Proposition 6.8. *Let the prior mean function μ^j be Lipschitz continuous, and the covariance kernel k^j be jointly Lipschitz continuous in its two variables, for all components j of the disturbance function $d(x)$. Then the disturbance bound $\hat{\mathcal{D}}(x)$, as defined in* (6.12)*, satisfies Assumptions 6.1 and 6.2.*

Proof. Assumption 6.1 holds independently of the Lipschitz condition. The bound $\hat{\mathcal{D}}(x)$ given by (6.12) is a compact convex set in $\mathcal{D}$. As a result, the retraction map $\pi_x : \mathcal{D} \to \hat{\mathcal{D}}(x)$ assigning every $d \in \mathcal{D}$ its (unique) nearest point in $\hat{\mathcal{D}}(x)$ is a Lipschitz continuous function (with Lipschitz constant equal to 1); of course with $\pi_x(\hat{d}) = \hat{d}$ for all $\hat{d} \in \hat{\mathcal{D}}(x)$. Then, the function $H_x(d, t) := (1 - t)d + t\pi_x(d)$ is continuous by composition and further satisfies $H_x(d, 0) = d, H_x(d, 1) \in \hat{\mathcal{D}}(x), H_x(\hat{d}, 1) = \hat{d}$ for all $d \in \mathcal{D}$ and $\hat{d} \in \hat{\mathcal{D}}(x)$.

Assumption 6.2 can be shown to hold by noting that the extrema of each of the intervals in (6.12) are affine in $\bar{d}^j(x)$ and $\sigma^j(x)$, which are Lipschitz continuous in x by Proposition 6.7. This implies that the position of all vertices of the hyperrectangle in (6.12) varies as a Lipschitz continuous function of x, and so does, as a result, the nearest point in $\hat{\mathcal{D}}(x)$ to any fixed $d \in \mathcal{D}$. The map $r(x, d) = \pi_x(d)$ is hence Lipschitz continuous in x. Finally, since π_x is Lipschitz continuous in d, r is also uniformly continuous in d. □

Finally, we can show that, under the same Lipschitz assumptions on the Gaussian process prior, the disturbance bound $\hat{\mathcal{D}}(x)$ is Lipschitz continuous under the Hausdorff distance, which we required in Proposition 6.1.

Proposition 6.9. *Let the prior mean function μ^j be Lipschitz continuous, and the covariance kernel k^j be jointly Lipschitz continuous in its two variables, for all components j of the disturbance function $d(x)$. Then the set-valued map $\hat{\mathcal{D}}$ is Lipschitz continuous under the Hausdorff distance.*

Proof. Since the disturbance set $\hat{\mathcal{D}}(x)$ given by (6.12) is a hyperrectangle, the Hausdorff distance between the disturbance sets $\hat{\mathcal{D}}(x_1)$ and $\hat{\mathcal{D}}(x_2)$ is upper-bounded by the maximum distance between any pair of corners:

$$\delta(x_1, x_2) := d_H\big(\hat{\mathcal{D}}(x_1), \hat{\mathcal{D}}(x_2)\big) \leq \max_i \max_k |c_i - c_k| \ ,$$

with c_i,c_k used to enumerate all corners of each of the two hyperrectangles. For simplicity of exposition, we use the equivalence of all norms in $\mathbb{R}^{d_n}$ to upper-bound the above, arbitrary norm in $\mathbb{R}^{n_d}$, by the infinity norm, up to a constant factor m, which in combination with (6.12) gives:

$$\delta(x_1, x_2) \leq m \cdot \max_j \left(|\bar{d}^j(x_1) - \bar{d}^j(x_2)| + |z\sigma^j(x_1) - z\sigma^j(x_2)|\right).$$

Now, by Proposition 6.7, $\bar{d}^j(x)$ and $\sigma^j(x)$ are Lipschitz continuous in x; let their respective constants be L^j_μ and L^j_σ. We then have

$$d_H\big(\hat{\mathcal{D}}(x_1), \hat{\mathcal{D}}(x_2)\big) \leq m \cdot \max_j \left(L^j_\mu + zL^j_\sigma\right)|x_1 - x_2| \ ,$$

which proves Hausdorff Lipschitz continuity of the set-valued map $\hat{\mathcal{D}}$, with a Lipschitz constant $L_{\hat{\mathcal{D}}}$ upper bounded by $m \cdot \max_j L^j_\mu + zL^j_\sigma$. □

Chapter 7

Confidence-Aware Planning with Human Models

> It's not who I am underneath,
> but what I do, that defines me.
>
> Bruce Wayne
> *Batman Begins*, 2005

This chapter is based on the papers "Probabilistically Safe Robot Planning with Confidence-Based Human Predictions" [19], "Confidence-Aware Motion Prediction for Real-Time Collision Avoidance" [174], and "A Scalable Framework for Real-Time Multi-Robot, Multi-Human Collision Avoidance" [20], written in collaboration with Andrea Bajcsy, David Fridovich-Keil, Sylvia Herbert, Steven Wang, Claire Tomlin, and Anca Dragan.

One of the fundamental challenges facing modern robotic technologies as their reach continues to expand beyond controlled factory environments is correct and safe operation in the midst of human beings. While "robot accidents" involving humans do occasionally take place in industrial settings [175], most of them are due to either malfunctioning equipment or human operator error, often by unwittingly entering the restricted workspace of an operational robot. In contrast, many of the potential and already-emerging new applications of robotics require them to operate in an unrestricted space shared with people. For example, it is possible today to find mobile robotic systems providing security and surveillance services in public spaces like office buildings and malls, not without incidents [176, 177]. As discussed in Chapter 4, consumer drones are being increasingly equipped with autonomous functionalities [100, 101], including the ability to follow their users while taking pictures or video footage of them.

In all of these scenarios, the safety of the robotic system will typically include requirements like avoiding physical collisions with nearby humans, and as a result will not depend exclusively on the decisions of the robotic system, but also on the actions taken by the

humans in question. Therefore, in order to competently ensure safety, the system needs to reason predictively about the relevant human behavior. Unfortunately, much like the physical dynamics that were the focus of Chapter 6, the behavior of human agents can be extremely difficult to predict accurately.

While it would in principle be possible to apply the machinery in Chapter 6 to the "uncertain dynamics" of human agents, the particular complexity of human decision-making grants a separate treatment. On the one hand, the degree of uncertainty that a robot may have about the future behavior of a human may often be much larger than, say, about its own physical motion. This means that attempting a bounded worst-case analysis may prove impractical, essentially leading us to the conclusion that the human "might do anything" and therefore almost every course of action is potentially unsafe—even if safe solutions are found, the resulting robotic operation may be exceedingly conservative, to the point of preventing it from executing its task altogether. On the other hand, human behavior differs from most physical processes in that it tends to be highly purpose-driven. As a result of this, it presents a considerable degree structure that lends itself to *teleological* analysis more easily than to *mechanistic* analysis.

This chapter therefore focuses on safe decision-making for robotic systems in environments shared with human beings. It considers the *reality gap* problem in the specific case of attempting to model and predict the uncertain behavior of people, and it introduces a method for the robotic system to adapt its decisions in response to the reliability of its human model in light of ongoing observations.

One popular predictive approach to anticipate human motion is to model humans as approximately rational with respect to an objective function learned from prior data [178, 179]. When a person is moving in accordance with the learned objective (e.g. to a known goal location), these models often make accurate predictions and the robot can easily find a safe path around the person. Unfortunately, in practice all such models of human intent are inevitably limited in their expressive power, and the robot's model of the human will not be able to capture all possible movements that it might eventually observe. For example, the human might walk toward another goal location that the robot does not know about, or move to avoid an obstacle of which the robot is unaware. In these cases where the human's motion diverges from the model's predictions, safety might be compromised. In Figure 7.1 (left), the robot fails to reason about the human avoiding the unobserved obstacle and gets dangerously close to the human.

One method to mitigate the effects of model inaccuracy is for the robot to re-compute its human model over time. However, restrictions in sensing and in the availability of human data limit how much a model can be refined online without overfitting. Alternatively, the robot can reason about its confidence in its current model's predictions. This chapter introduces a method in which the robot continually estimates its confidence in its human model in real time and adapts its motion plan according to this confidence (Figure 7.1, right). In particular, our approach leverages the so-called "rationality" coefficient in the commonly used Boltzmann model of approximately rational human behavior [79, 180] as a time-varying indicator of the model's predictive performance. This is a single scalar parameter that can be

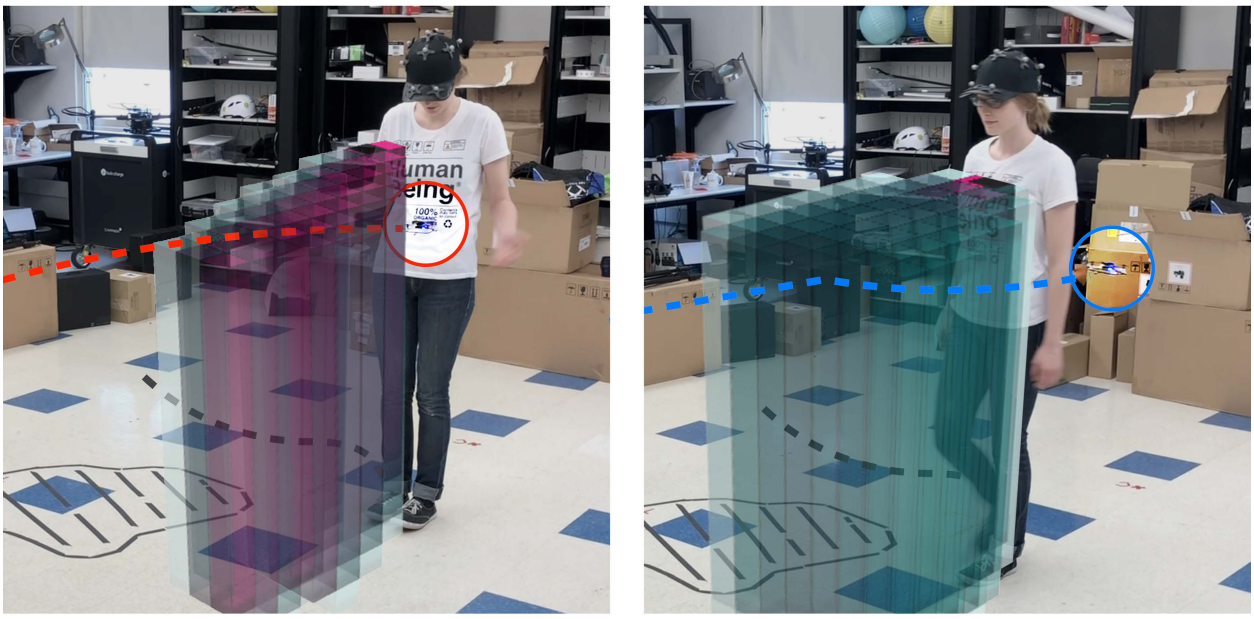


Figure 7.1: When planning around humans, predictive models can enable robots to reason about future motions the human might take. These predictions rely on human motion models, but such models will often be incomplete and lead to inaccurate predictions and even collisions (left). Our method addresses this by updating its *human model confidence* in real time (right). Video: `https://youtu.be/2ZRGxWknENg`

tractably inferred at deployment time. We couple the resulting confidence-aware human motion predictions with a provably safe motion planner to obtain *probabilistically safe* robotic motion plans that are conservative when appropriate but efficient when possible.

This chapter makes two key contributions: (1) a real-time Bayesian framework for reasoning about the uncertainty inherent in a model's prediction of human movement, and (2) extending a state-of-the-art, provably safe, real-time robotic motion planner to incorporate our time-varying, probabilistic human predictions. Together, these two contributions facilitate the real-time generation of robot trajectories through human-occupied spaces. Further, they guarantee that when the robot tracks these trajectories at run-time they will be collision-free with arbitrarily high probability.

Related Work

One common approach for predicting human actions is supervised learning, where the current state and the history of the human's actions are used directly to predict future actions. Such approaches have enabled inference and planning around human arm motion [181–185],

navigation [182], plans for multi-step tasks like assembly [185], and driving [186].

Rather than predicting actions directly, an alternative is for the robot to model the human as a rational agent seeking to maximize an unknown objective function. The human's actions up to a particular time may be viewed as evidence about this objective from which the robot may infer the parameters of that objective. Assuming that the human seeks to maximize this objective in the future, the robot can predict her future movements [76, 180]. In this chapter, we build on in this work by introducing a principled online technique for estimating confidence in such a learned model of human motion.

Once armed with a predictive model of the human motion, the robot needs to generate real-time trajectory plans that will both be dynamically feasible and safely avoid future collisions. A more detailed overview of the robotic motion planning literature can be found in Chapter 5, and multi-agent planning methods are discussed in Chapter 4. While some of the geometric motion planning methods in the robotics literature have been combined with probabilistically moving obstacles [178, 187], they do not consider the endogenous dynamics of the robot or exogenous disturbances such as wind. As a result, the robot may deviate from the planned path and potentially collide with obstacles, as we have seen in detail in Chapter 5. The approach presented in this chapter builds upon the robust motion planning framework from Chapter 5 to safely and dynamically navigate around uncertain moving obstacles in real time.

7.1 Safe Robot Trajectories under Uncertain Human Motion

We consider a single robot moving to a preset goal location in a space shared with a single human, and assume that the human expects the robot to avoid her. Therefore, it is the robot's responsibility to maintain a safe distance from the human at all times. We present our theory for a general single human and single robot setting, and use the running example of quadrotor navigating around a walking human to illustrate the proposed approach and demonstrate the utility of our method.

7.1.1 Motion Model

Let the state of the human be $x_H \in \mathbb{R}^{n_H}$, where n_H is the dimension of the human state space. We similarly define the robot's state, for planning purposes, as $x_R \in \mathbb{R}^{n_R}$. These states could represent the positions and velocities of a mobile robot and a human in a shared environment or the kinematic configurations of a human and a robotic manipulator in a common workspace. The human and robot are each modeled by their dynamics:

$$\dot{x}_H = f_H(x_H, u_H) \qquad \dot{x}_R = f_R(x_R, u_R) \tag{7.1}$$

where $u_H \in \mathbb{R}^{m_H}$ and $u_R \in \mathbb{R}^{m_R}$ are the control actions of the human and robot, respectively.

The robot ultimately needs to plan and execute a trajectory to a goal state according to some notion of efficiency, while avoiding collisions with the environment (encoded through static failure states $\mathcal{F}_R^0 \subset \mathbb{R}^{n_R}$) or the human. We define the danger zone $\mathcal{Z}_{H,R} \subset \mathbb{R}^{n_H} \times \mathbb{R}^{n_R}$ as the set of joint robot-human states to be avoided, e.g. because they imply physical collisions. To avoid reaching this set, the robot must reason about the human's future motion when constructing its own motion plan.

Example 7.1. *We introduce a running example for illustration throughout the chapter. In this example we consider a small quadrotor vehicle that needs to fly to targets $\mathcal{T}_R \subset \mathbb{R}^3$ in a room where a human is walking. For the purposes of planning, the quadrotor's state is given by its position in space $r_R = [rx_R, \mathrm{y}_R, \mathrm{z}_R]$, evolving as*

$$\begin{bmatrix} \dot{rx}_R \\ \dot{\mathrm{y}}_R \\ \dot{\mathrm{z}}_R \end{bmatrix} = \begin{bmatrix} v_{\mathrm{x},R} \\ v_{\mathrm{y},R} \\ v_{\mathrm{z},R} \end{bmatrix} \quad , \tag{7.2}$$

with velocity controls assumed decoupled in each spatial direction: $v_{\mathrm{x},R}, v_{\mathrm{y},R}, v_{\mathrm{z},R} \in [-\bar{v}_R, \bar{v}_R]$, where $\bar{v}_R = 0.25$ m/s. The human can only move by walking and therefore her state is given by planar coordinates $r_H = [rx_H, \mathrm{y}_H]$ evolving through simple motion dynamics

$$\begin{bmatrix} \dot{rx}_H \\ \dot{\mathrm{y}}_H \end{bmatrix} = v_H \begin{bmatrix} \cos\psi_H \\ \sin\psi_H \end{bmatrix} \quad . \tag{7.3}$$

At any given time, the human is assumed to either move at a leisurely walking speed ($v_H \approx 1$ m/s) or remain still ($v_H \approx 0$).

In this example, $\mathcal{Z}_{H,R}$ consists of joint robot-human states in which the quadrotor is flying within a square of side length $L_c = 0.3$ m centered around the human's location, while at any altitude:

$$\mathcal{Z}_{H,R} := \left\{ (r_R, r_H) \in \mathbb{R}^5 : \|(rx_R, \mathrm{y}_R) - (rx_H, \mathrm{y}_H)\|_\infty < \frac{L_c}{2} \right\} \tag{7.4}$$

The robot is must also avoid a static failure set $\mathcal{F}_R^0 \subset \mathbb{R}^3$ comprised by all positions outside of a box with a square base of side $L = 3.66$ m and height $H = 2$ m, representing the physical dimensions of the room. The human is modeled as being similarly constrained to remain in a square of the same dimensions.

7.1.2 Robot Dynamics

Ideally, robots should plan their motion based on a high-fidelity model of their dynamics, accounting for inertia, actuator limits, and environment disturbances. Unfortunately, reasoning with such complex models is computationally prohibitive in most practical cases.As

a result, the models used for planning typically constitute a simplified representation of the physical dynamics of the real robot, and are therefore subject to some error that can have critical implications for safety. In particular, let $s_R \in \mathbb{R}^{n_S}$ denote the state of the robot in the higher-fidelity dynamical model, and let $\pi : \mathbb{R}^{n_S} \to \mathbb{R}^{n_R}$ be a known function that projects this higher-fidelity state onto a corresponding planning state, i.e $x_R = \phi(s_R)$. [1] A planner which operates on x_R may generate a trajectory which is difficult to track or even infeasible under the more accurate dynamical model. Thus reasoning with the planning model alone is not sufficient to guarantee safety for the real robot.

Example 7.2. *We model our quadrotor with the following flight dynamics (in near-hover regime):*

$$\begin{bmatrix} \dot{\mathrm{x}} \\ \dot{\mathrm{y}} \\ \dot{\mathrm{z}} \end{bmatrix} = \begin{bmatrix} v_{\mathrm{x}} \\ v_{\mathrm{y}} \\ v_{\mathrm{z}} \end{bmatrix} \quad , \qquad \begin{bmatrix} \dot{v}_{\mathrm{x}} \\ \dot{v}_{\mathrm{y}} \\ \dot{v}_{\mathrm{z}} \end{bmatrix} = \begin{bmatrix} g \tan\theta \\ -g \tan\phi \\ \tau - g \end{bmatrix} \quad , \tag{7.5}$$

where $[\mathrm{x}, \mathrm{y}, \mathrm{z}]$ *is the quadrotor's position in space and* $[v_{\mathrm{x}}, v_{\mathrm{y}}, v_{\mathrm{z}}]$ *is its velocity expressed in the fixed world frame, with thrust* τ *and attitude angles (roll* ϕ *and pitch* θ*) as controls. The quadrotor's motion planner generates nominal kinematic trajectories in the lower-dimensional* $[\mathrm{x}, \mathrm{y}, \mathrm{z}]$ *position state space. Therefore we have a linear projection map* $\phi(s_R) = [I_3, 0_3]s_R$*, that is,* x_R *retains the position variables in* s_R *and discards the velocities.*

7.1.3 Predictive Human Model

The robot has a predictive model of the human's motion, based on a set of parameters whose values may be inferred under a Bayesian framework or otherwise estimated over time. Extensive work in econometrics and cognitive science has shown that human behavior can be well modeled by utility-driven optimization [74, 180, 188]. Thus, the robot models the human as optimizing a reward[2] function, $L_H(x_H, u_H; \theta)$, that depends on the human's state and action, as well as a set of parameters θ. This reward function could be a linear combination of features as in many inverse optimal control implementations (where the weighting θ between the features needs to be learned), or more generally learned through function approximators such as deep neural networks (with θ as the trained network weights) [189].

We assume that the robot has a suitable human reward function, either learned offline from prior human demonstrations or otherwise encoded by the system designers. With this, the robot can compute the human's policy as a probability distribution over actions conditioned on the state. Using maximum-entropy assumptions [79] and inspiration from

[1] Note that this is a special case of the more general relative relation between the planning and tracking dynamics in Chapter 5.

[2] In the optimal control terminology introduced in Chapter 2, Section 2.2, this reward corresponds to a Lagrangian objective.

noisy-rationality models used in cognitive science [180], the robot models the human as more likely to choose actions with high expected utility, in this case the state-action value (or Q-value):

$$P(u_H^t \mid x_H^t; \beta, \theta) = \frac{e^{\beta Q_H(x_H^t, u_H^t; \theta)}}{\sum_{\tilde{u}} e^{\beta Q_H(x_H^t, \tilde{u}; \theta)}} \ . \tag{7.6}$$

Example 7.3. *The quadrotor's model of the human assumes that she intends to reach some target location $g \in \mathbb{R}^2$ in the most direct way possible. The human's reward function is given by the distance traveled $L_H(x_H, u_H; g) = -||u_H||_2$ and human trajectories are constrained to terminate at g. The state-action value, parametrized by $\theta = g$, captures the optimal cost of reaching g from x_H when initially applying u_H: $Q_H(x_H, u_H; g) = -||u_H||_2 - ||x_H + u_H - g||_2$.*

The coefficient β is traditionally called the *rationality coefficient* and it determines the degree to which the robot expects to observe human actions aligned with its model of utility. A common interpretation of $\beta = 0$ is a human who appears "irrational," choosing actions uniformly at random and completely ignoring the modeled utility, while $\beta \to \infty$ corresponds a "perfectly rational" human. Instead, we believe that β can be given a more pragmatic interpretation related to the accuracy with which the robot's model of the human is able to explain her motion. Consistently, in this chapter, we refer to β as *model confidence*.

Note that we assume the human does not react to the robot. This assumption can realistically capture plausible shared-space settings in which lightweight robots (e.g. micro-drones) may be expected to carry out services such as indoor surveillance in a building while minimizing interference with human activity. Additionally, to the extent that a more compliant human will tend to avoid collisions with the robot, the robot may still benefit in such scenarios—it is merely not assuming any cooperation *a priori* in its planning.

7.1.4 Probabilistic Safe Motion Planning Problem

The problem that the robot needs to solve is to plan a trajectory that, when tracked by the physical system, will reach a goal state as efficiently as possible while avoiding collisions with high confidence, based on an informed prediction of the human's future motion.

Since any theoretical guarantee is tied to the model it is based on, safety guarantees will inherit the probabilistic nature of human predictions. This induces a fundamental tradeoff between safety and *liveness*: predictions of human motion may assign non-zero probability to a wide range of states at a future time, which may severely impede the robot's ability to operate in the shared space with "absolute safety" (only absolute according to the model). Therefore, depending on the context, the designers or operators of the system will need to determine what is an acceptable probability that a robot's plan will conflict with the human's future motion. Based on this, the robot's online planning algorithm will determine when a

motion plan is predicted to be *sufficiently safe*. In our demonstrated system, we use a 1% collision probability threshold for planning.

Our goal now is to find efficient robot motion plans that will keep collisions with a human below an acceptable probability. Formally, given a current state $x_R^{\text{now}} \in \mathbb{R}^{n_R}$, a cumulative cost $c : \mathbb{R}^{n_R} \times \mathbb{R}^{m_R} \to \mathbb{R}$, a probability threshold $P_{\text{th}} \in [0, 1]$ and a final time T, we define the constrained planning problem:

$$\min_{u_R^{t:T}} \sum_{\tau=t}^{T} c(x_R^\tau, u_R^\tau) \tag{7.7a}$$

$$\text{s.t. } x_R^t = x_R^{\text{now}} \tag{7.7b}$$

$$x_R^{\tau+1} = \tilde{f}_R(x_R^\tau, u_R^\tau), \quad \tau \in t, ..., T-1 \tag{7.7c}$$

$$x_R^{\tau+1} \notin \mathcal{F}_R^0 \quad \tau \in t, ..., T-1 \tag{7.7d}$$

$$P_{\text{coll}}^{t:T} := P\big(\exists \tau \in \{t, ..., T\} : \text{coll}(x_R^\tau, x_H^\tau)\big) \leq P_{\text{th}} \tag{7.7e}$$

with $\tilde{f}_R$ a discrete-time approximation of the dynamics f_R. The term $\text{coll}(x_R^t, x_H^t)$ is a Boolean variable indicating whether the human and the robot are in collision.

The safety analysis necessary to solve this online motion planning problem therefore has two main components, the robot's state and the human's state, both of which are affected by uncertainty in their evolution over time. We tackle these two sources of uncertainty through a combined method that draws simultaneously on the two main approaches to uncertain systems: probabilistic and worst-case analysis.

Example 7.4. *The quadrotor's cost can be a weighted combination of distance traversed and time elapsed on its way to a specified goal:* $c(x_R, u_R) = ||u_R||_2 + c_0$.

The proposed approach in this chapter follows two central steps to provide a quantifiable, high-confidence collision avoidance guarantee for the robot's motion around the human. In Section 7.2 we present our proposed Bayesian framework for reasoning about the uncertainty inherent in a model's prediction of human behavior. Based on this inference, we demonstrate how to generate a real-time probabilistic prediction of the human's motion over time. Next, in Section 7.3 we introduce a theoretical extension to a state-of-the-art, provably safe, real-time robotic motion planner to incorporate our time-varying probabilistic human predictions yielding a quantitative probabilistic safety certificate.

7.2 Confidence-Aware Human Motion Prediction

Predictions of human motion, even when based on well-informed models, may eventually perform poorly when the human's behavior outstrips the model's predictive power. Such situations can have a negative impact on safety if the robot fails to appropriately, and quickly, notice the degradation of its predictions.

It will often be the case in practice that the same model will perform variably well over time in different situations and for different people. In some cases this model might be perfectly representative, in others the robot might not have access to some important feature that explains the human's behavior, and therefore the robot's conservativeness should vary accordingly.

Given a utility-based human model in the form of (7.6), the inverse temperature parameter β can be leveraged as an indicator of the model's predictive ability. In particular, the entropy of the human control distribution in (7.6) is a decreasing function of β: high values of β place more probability mass on high-utility control actions u_H, whereas low values of β spread the probability mass more evenly between different control inputs, regardless of the modeled utility Q_H. Therefore, β naturally quantifies how well the human's motion is expected to agree with the notion of optimality encoded in Q_H. While the commonly used term "rationality coefficient" seems to imply that discrepancies between the two indicate a failure on the human's part to make the "correct" decisions, we argue that these inevitable disagreements are primarily a result of the model's inability to fully capture the human's behavior, and therefore refer to β as *model confidence*. By reasoning about this parameter over time, the robot can dynamically adapt its predictions (and therefore its motion plan) to the current reliability of its human model, resulting in more "introspective" confidence-aware autonomous behavior.

7.2.1 Real-time Inference of Model Confidence

Since the predictive performance of the model might change over time as the human's behavior evolves, we treat β as a hidden state in a hidden Markov model (HMM) framework, rather than a static parameter. We can maintain a Bayesian *belief* about the possible values of β, starting with a prior probability distribution b^0_- over the initial value of β and updating it at each new time $t = 1, 2, \cdots$ given measurements of the human's state and actions.

The hidden state may evolve between subsequent time steps, accounting for the important fact that the predictive accuracy of the human model may change over time as unmodeled factors in the human's behavior become more or less relevant (for example, the human may change her mind unexpectedly, or react suddenly to some aspect of the environment that the robot is unaware of). Since by definition we do not have access to a model of these factors, we use a naive "ϵ-static" transition model: at each time t, β may, with some probability ϵ, be re-sampled from the initial distribution b^0_-, and otherwise retains its previous value. We define the belief over the next value of β (denoted by β') as the expectation of the conditional probability $P(\beta' \mid \beta)$, i.e. $b^t_-(\beta') := \mathbb{E}_{\beta \sim b^{t-1}}[P(\beta' \mid \beta)]$. Concretely, this expectation may be computed as

$$b^t_-(\beta') = (1-\epsilon) b^{t-1}(\beta') + \epsilon b^0_-(\beta') \ , \tag{7.8}$$

where $b^{t-1}(\beta) = P(\beta | x_H^{0:t-1}, u_H^{0:t-1})$ for $t \in \{1, 2, ...\}$ is the belief over β at time $t-1$ based on all observations made up until this time.

At every time step t, the robot obtains a new measurement of the human's action, u_H^t.[3] This measurement can be used as evidence to update the robot's belief $b^t(\cdot)$ about β over time via a Bayesian update, starting from the predicted distribution $b^t_-(\cdot)$:

$$b^t(\beta) = \frac{P(u_H^t \mid x_H^t; \beta, \theta) b^t_-(\beta)}{\sum_{\hat{\beta}} P(u_H^t \mid x_H^t; \hat{\beta}, \theta) b^t_-(\hat{\beta})} \ , \tag{7.9}$$

with $P(u_H^t | x_H^t; \beta, \theta)$ given by (7.6). It is critical to be able to perform this update extremely fast, which would be difficult to do in the original continuous hypothesis space $\beta \in [0, \infty)$ or even a large discrete set. Fortunately, as we will see in Section 7.4, maintaining a real-time Bayesian belief over a relatively small set of β values ($N_\beta \approx 10$ on a log-scale) achieves significant improvement relative to maintaining a fixed precomputed value.

On the other hand, the "ϵ-static" transition model (7.8) leads to the desirable consequence that old observations of the human's actions have a smaller influence on the current model confidence distribution than recent observations. In fact, if no new observations are made, successively applying time updates asymptotically contracts the belief towards the initial distribution, that is, $b^t_-(\cdot) \to b^0_-(\cdot)$. The choice of parameter ϵ effectively controls the rate of this contraction, with higher ϵ leading to more rapid contraction.

7.2.2 Human motion prediction

We can now use the belief over β to recursively propagate the human's motion over time and obtain a probabilistic prediction of her state at any number of time steps into the future. In particular, suppose that we know the probability that the human is in each state x_H^τ at some future time step τ. We know that (according to our utility model) the probability of the human choosing control u_H^τ if she is in state x_H^τ is given by (7.6). Accounting for the otherwise deterministic dynamics model $\tilde{f}_H$, we obtain the following expression for the human's state distribution at the following time step $\tau + 1$:

$$\begin{aligned} P(x_H^{\tau+1}; \beta, \theta) = \sum_{x_H^\tau, u_H^\tau} & P(x_H^{\tau+1} \mid x_H^\tau, u_H^\tau; \beta, \theta) \cdot \\ & P(u_H^\tau \mid x_H^\tau; \beta, \theta) P(x_H^\tau; \beta, \theta) \ , \end{aligned} \tag{7.10}$$

for a particular choice of β; for the deterministic dynamics in our case, $P(x_H^{\tau+1} \mid x_H^\tau, u_H^\tau; \beta, \theta) = \mathbb{1}\{x_H^{\tau+1} = \tilde{f}_H(x_H^\tau, u_H^\tau)\}$.

Taking the posterior expectation over β according to our belief at the current step time t, we obtain the overall occupancy probability distribution at each future time step τ:

$$P(x_H^\tau; \theta) = \mathbb{E}_{\beta \sim b^t} P(x_H^\tau; \beta, \theta) \ . \tag{7.11}$$

[3] In practice, the robot measures the evolution of the human state and computes the associated action by inverting the motion model.

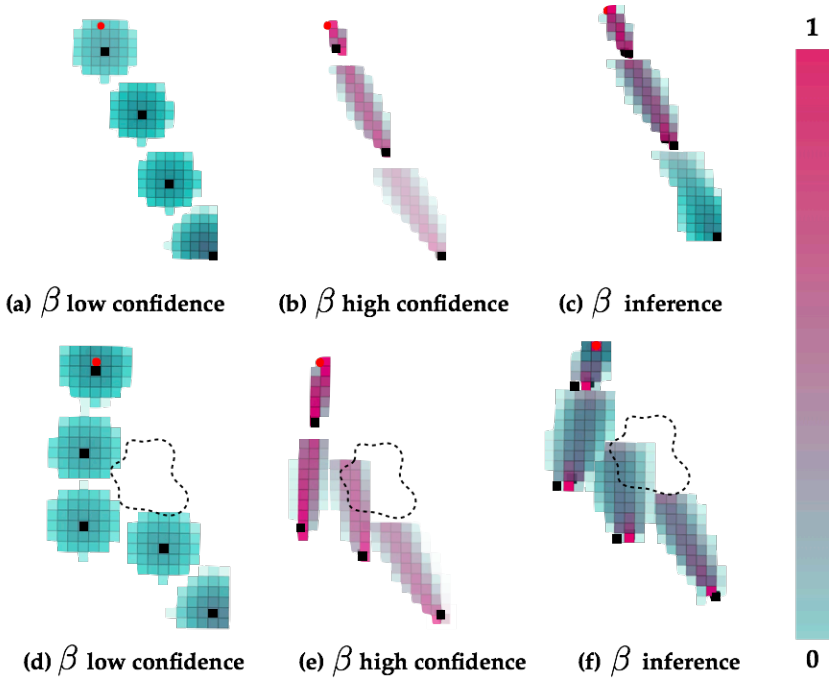


Figure 7.2: Snapshots of human trajectory and probabilistic model predictions in the unmodeled obstacle scenario. *Top row:* Human moves from the bottom right to a goal marked as a red circle. *Bottom row:* Human changes course to avoid a spill on the floor. The first two columns show the predictions for low and high model confidence; the third column shows the predictions for Bayesian model confidence. Video: `https://youtu.be/lh_E9rW-MJo`

Example 7.5. *The simplest scenario in our running example involves a human moving towards a known goal. In Figure 7.2(a-c), the human acts predictably, moving directly to the goal. Each subfigure shows the robot's human prediction under different confidence conditions. Predictions for the second scenario, where the human deviates from her path to avoid a coffee spill on the ground, are shown in Figure 7.2(d-f).*

7.2.3 Integrating Model Confidence into Online Model Updates

When a robot is faced with human behavior that is not well explained by its current model, it can attempt to update some of its elements to better fit the observed human actions. These elements can include parameters, hyperparameters, or potentially even the structure of the model itself. Assuming that the parameters can be tractably adjusted online, this update may result in better prediction performance.

Even under online model updates, it continues to be necessary for the robot to reason about model confidence. In this section we demonstrate how reasoning about model confidence can be done compatibly (and in some cases jointly) with model parameter updates.

Recall that θ denotes the set of parameters in the human's utility model. The ideal approach is to perform inference over both the model confidence, β, and the model parameters, θ by maintaining a joint Bayesian belief, $b^t(\beta, \theta)$. The joint Bayesian belief update rule takes the form

$$b^t(\beta,\theta) = \frac{P(u_H^t \mid x_H^t; \beta, \theta) b_-^t(\beta,\theta)}{\sum_{\hat{\beta},\hat{\theta}} P(u_H^t \mid x_H^t; \hat{\beta}, \hat{\theta}) b_-^t(\hat{\beta},\hat{\theta})} \ , \tag{7.12}$$

with $b^t(\beta, \theta) = P(\beta, \theta \mid x_H^{0:t}, u_H^{0:t})$.[4] This approach can be practical for parameters taking finitely many values from a discrete set, for example, possible distinct modes for a human driver (distracted, cautious, aggressive).

Example 7.6. *The quadrotor's model of the human considers a number of known frequently-visited locations $\theta \in \{g_1, ..., g_N\}$ that she might intend to walk to next. However, there may be additional unmodeled destinations, or more complex objectives driving the human's motion in the room (for example, she could be searching for a misplaced object, or pacing while on the phone). Figure 7.3 shows how reasoning about model confidence as well as the human's destination enables the robot to navigate confidently while the human's motion is well explained by the model, and automatically become more cautious when it departs from its predictions. More detailed results are presented in Section 7.4.*

For certain scenarios or approaches it may not be practical to maintain a full Bayesian belief on the parameters, and these are instead estimated over time (for example, through a maximum likelihood estimator (MLE), or by shallow re-training of a pre-trained neural network). In these cases, a practical approach can be to maintain a "bootstrapped" belief on β by running the Bayesian update on the running parameter estimate $\bar{\theta}$:

$$\bar{b}^t(\beta) = \frac{P(u_H^t \mid x_H^t; \beta, \bar{\theta}^t) \bar{b}_-^t(\beta)}{\sum_{\hat{\beta}} P(u_H^t \mid x_H^t; \hat{\beta}, \bar{\theta}^t) \bar{b}_-^t(\hat{\beta})} \ . \tag{7.13}$$

Example 7.7. *The quadrotor's predictions of human motion are parameterized by her walking speed v_H; the quadrotor maintains a simple running average based on recent motion-capture measurements, and incorporates the current estimate into inference and prediction.*

When it is not desirable or computationally feasible to update the parameter estimate $\bar{\theta}^t$ continually, model confidence can also be used as an indicator of when re-estimating these parameters may be most useful—namely as confidence in the model under the current parameter estimates degrades.

[4] Analogously to the case with β-only inference, the parameters θ can be allowed to evolve as a hidden state.

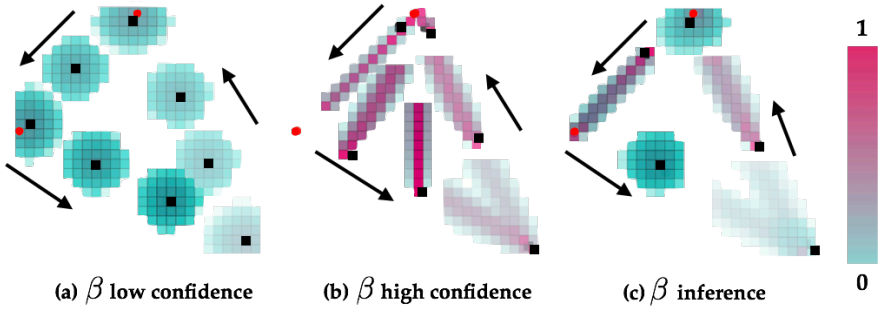


Figure 7.3: Human motion predictions under modeled and unmodeled goals. The human is moving in a counter-clockwise motion to two modeled goals (marked in red), and then to a third unmodeled goal (located at the same position as the start). Subfigures (a) and (b) show the predictions for a low and high confidence, respectively. Subfigure (c) shows the predictions using our inferred model confidence, where the robot is confident when the human is moving "rationally", and uncertain when the human behavior does not match the robot's model.

7.3 Safe Probabilistic Planning and Tracking

Once it can generate real-time probabilistic predictions of the human's motion, the robot needs to plan a trajectory that will, with high probability, avoid collisions with her. On the one hand, any rigorous safety analysis for a robotic system needs to account for deviations of the actual dynamic trajectory from the ideal motion plan. On the other hand, since human motion predictions are by nature uncertain, the safety analysis will necessarily be quantified in probabilistic terms. To this end, we build on the FaSTrack framework introduced in Chapter 5, which provided us with control-theoretic robust safety certificates in the presence of deterministic obstacles, and extend the theoretical analysis to provide probabilistic certificates allowing uncertain dynamic obstacles, in order to handle the uncertain future motion of human agents.

7.3.1 Robust Tracking, Probabilistic Safety

Recall that x_R and u_R are the robot's state and control input, for the purposes of motion planning. The FaSTrack framework uses Hamilton-Jacobi analysis to provide a simple real-time motion planner with a worst-case tracking error bound for the dynamic robot (and a tracking control policy to enforce the bound). This bound $\mathcal{E}$ is a trajectory tracking certificate that can be passed to the online planning algorithm for real-time safety verification: the dynamical robot is guaranteed to *always* be somewhere within the associated robust tracking set given by the Minkowski sum $\tilde{\mathcal{X}}(x_R) = \tilde{\mathcal{X}}(x_R) + \mathcal{E}$. Therefore the planner can generate safe plans by ensuring that the entire robust tracking set remains collision-free throughout the trajectory. Note that the planner only needs to know $\mathcal{E}$ and otherwise requires no explicit

understanding of the high-fidelity model.

Example 7.8. *Since dynamics* (7.5) *are decoupled in the three spatial directions, the bound* $\mathcal{E}$ *computed by FaSTrack is an axis-aligned box of dimensions* $\mathcal{E}_x \times \mathcal{E}_y \times \mathcal{E}_z$.

Unfortunately, planning algorithms for collision checking against deterministic obstacles cannot be readily applied to our problem. Instead, a trajectory's collision check should return the probability that it *might* lead to a collision. Based on this probability, the planning algorithm can discriminate between trajectories that are *sufficiently safe* and those that are not.

As discussed in Section 7.1, a safe online motion planner should continually check the probability that, at any future time τ, $(\pi(s_R^\tau), x_H^\tau) \in \mathcal{Z}_{H,R}$. The tracking error bound guarantee from FaSTrack allows us to conduct worst-case analysis on collisions given a human state x_H: if no point in the robust tracking set $\tilde{\mathcal{X}}(x_R)$ is in the collision set with x_H, we can guarantee that the robot is not in collision with the human.

The probability of a collision event for any point x_R^τ in a candidate trajectory plan, assuming worst-case tracking error, can be computed as the total probability that x_H^τ will be in collision with *any* of the possible robot states $\tilde{x}_R \in \{x_R^\tau + \mathcal{E}\}$. For each robot planning state $x_R \in \mathbb{R}^{n_R}$ we can define the set of human states in potential collision with the robot:

$$\mathcal{H}_H(x_R) := \{\tilde{x}_H \in \mathbb{R}^{n_H} : \exists \tilde{x}_R \in \tilde{\mathcal{X}}(x_R), (\tilde{x}_R, \tilde{x}_H) \in \mathcal{Z}_{H,R}\} \ . \tag{7.14}$$

The following result is then true by construction.

Proposition 1: *The probability of a robot with worst-case tracking error* $\mathcal{E}$ *being in collision with the human at any trajectory point* x_R^τ *is bounded above by the probability mass of* x_H^τ *contained within* $\mathcal{H}_H(x_R^\tau)$.

Therefore, the left-hand side of the inequality in our problem's safety constraint (7.7e) can be rewritten as

$$P_{\text{coll}}^{t:T} = 1 - \prod_{\tau=t}^{T} P\big(x_H^\tau \notin \mathcal{H}_H(x_R^\tau) \mid x_H^\tau \notin \mathcal{H}_H(x_R^s), t \le s < \tau\big). \tag{7.15}$$

Evaluating the above probability exactly would require reasoning jointly about the distribution of human states over all time steps, or equivalently over all time trajectories $x_H^{0:T}$ that the human might follow. Due to the need to plan in real time, we must in practice approximate this distribution.

Since assuming independence of collision probabilities over time is both unrealistic and overly conservative, we instead seek to find a tight lower bound on a trajectory's overall collision probability based on the marginal probabilities at each moment in time. In particular, based on the positive correlation over time resulting from human motion continuity, we first consider replacing each conditional probability $P\big(x_H^\tau \notin \mathcal{H}_H(x_R^\tau) \mid x_H^s \notin \mathcal{H}_H(x_R^s), t \le s < \tau\big)$ by 1 for all $t > 0$. This would then give the lower bound

$$P_{\text{coll}}^{t:T} \ge 1 - P\big(x_H^t \notin \mathcal{H}_H(x_R^t)\big) = P\big(x_H^t \in \mathcal{H}_H(x_R^t)\big) \ , \tag{7.16}$$

which would seem like an unreasonably optimistic approximation. However, note that probabilities can be conditioned in any particular order (not necessarily chronological) and we can therefore generate $T - t + 1$ lower bounds of the form $P_{\text{coll}}^{t:T} \geq P\big(x_H^\tau \in \mathcal{H}_H(x_R^\tau)\big)$ for $\tau \in \{t, \dots, T\}$, again by replacing all successive conditional non-collision probabilities by 1. Taking the tightest of all of these bounds, we can obtain an informative, yet quickly computable, approximator for the sought probability:

$$P_{\text{coll}}^{t:T} \approx \max_{\tau \in \{t:T\}} P\big(x_H^\tau \in \mathcal{H}_H(x_R^\tau)\big) \ . \tag{7.17}$$

In other words, we are replacing the probability of collision of an entire trajectory with the highest marginal collision probability at *each point* in the trajectory. While this approximation will err on the side of optimism, we note that the robot's ability to continually replan as updated human predictions become available mitigates any potentially underestimated risks, since in reality the robot does not need to commit to a plan that was initially deemed safe, and will readily rectify as the estimated collision risk increases prior to an actual collision.

Example 7.9. *Given $\mathcal{Z}_{H,R}$ and $\mathcal{E}$, $\mathcal{H}_H(x_R^\tau)$ is the set of human positions within the rectangle of dimensions $(l + \mathcal{E}_x) \times (l + \mathcal{E}_y)$ centered on $[p_x^\tau, p_y^\tau]$. A human anywhere in this rectangle could be in collision with the quadrotor.*

7.3.2 Safe Online Planning under Uncertain Human Predictions

We can now use this real-time evaluation of collision probabilities to discriminate between valid and invalid trajectory candidates in the robot's online motion planning. Using the formulation in Section 7.2, we can quickly generate, at every time t, the marginal probabilities in (7.17) at each future time $\tau \in \{t, \dots, T\}$, based on past observations at times $0, \dots, t$. Specifically, for any candidate trajectory point x_R^τ, we first calculate the set $\mathcal{H}_H(x_R^\tau)$; this set can often be obtained analytically from (7.14), and can otherwise be numerically approximated from a discretization of $\mathcal{E}$. The planner then computes the instantaneous probability of collision $P\big(x_H^\tau \in \mathcal{H}_H(x_R^\tau)\big)$ by integrating $P\big(x_H^\tau \mid x_H^{0:t}\big)$ over $\mathcal{H}_H(x_R^\tau)$, and rejects the candidate point x_R^τ if this probability exceeds P_{th}.

Note that for search-based planners that consider candidate trajectories by generating a tree of timestamped states, rejecting a candidate node from this tree is equivalent to rejecting all further trajectories that would contain the node. This early rejection rule is consistent with the proposed approximation (7.17) of $P_{\text{coll}}^{t:T}$ while preventing unnecessary exploration of candidate trajecories that would ultimately be deemed unsafe.

As the robot is continuously regenerating its motion plan online as the human's predicted motion is updated, we simultaneously track the planned trajectory using our error feedback controller, which ensures that we deviate by no more than the tracking error bound $\mathcal{E}$. This planning and tracking procedure continues until the robot's goal has been achieved.

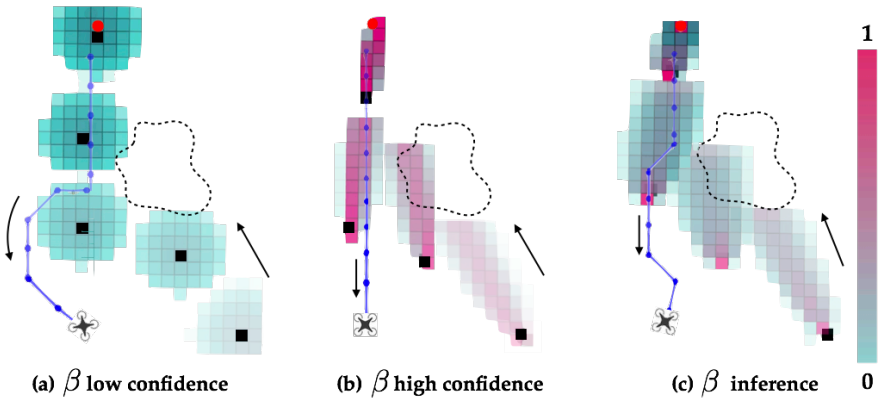


Figure 7.4: Scenario from Figure 7.2 visualized with robot's trajectory based on its current β. When β is low and the robot is not confident, it makes large deviations from its path to accommodate the human. When β is high, the robot refuses to change course and comes dangerously close to the human. With inferred model confidence, the robot balances safety and efficiency with a slight deviation around the human.

Example 7.10. *Our quadrotor is now required to navigate to a target position shown in Figure 7.2 without colliding with the human. Our proposed algorithm successfully avoids collisions at all times, replanning to leave greater separation from the human whenever her motion departs from the model. In contrast, robot planning with fixed model confidence is either overly conservative at the expense of time and performance or overly aggressive at the expense of safety.*

7.3.3 Connection to Reachability Analysis

We could conceivably require the robot's motion plans to avoid the set of all states the human would be physically capable of reaching at some time τ in the future, i.e. her forward-reachable set at time τ under the dynamical model f_H:

$$\mathcal{R}_F(x_H^t;\tau) = \{x : \exists u_H^t, \ldots, u_H^{\tau-1} : x_H^\tau = x\} \ . \tag{7.18}$$

This criterion can be attractive in that would come with a robust safety guarantee only contingent on the physical limits assumed on the humans motion (e.g. maximum speed) and not on our ability to anticipate her decision-making. Unfortunately, of course, generating a single open-loop motion plan that will stay clear of all physically realizable future human configurations is impractically conservative, if not altogether impossible, in most realistic scenarios.

The probabilistic predictions used in this chapter provide a less conservative approach by effectively allowing the robot to discard certain candidate future human states that, while

physically possible, are deemed extremely unlikely. Note that, by definition, the forward-reachable set always contains the support of any probability distribution over the human's future state at time τ; in the case of Boltzmann decision models, where all human actions have nonzero probability for finite Q_H and β, we in fact have that $\mathcal{R}_F(x_H^t;\tau)$ is always equal to the support of the probability distribution over x_H^t *regardless* of the Boltzmann model used. This is illustrated by Figure 7.5 for an example with a human pedestrian.

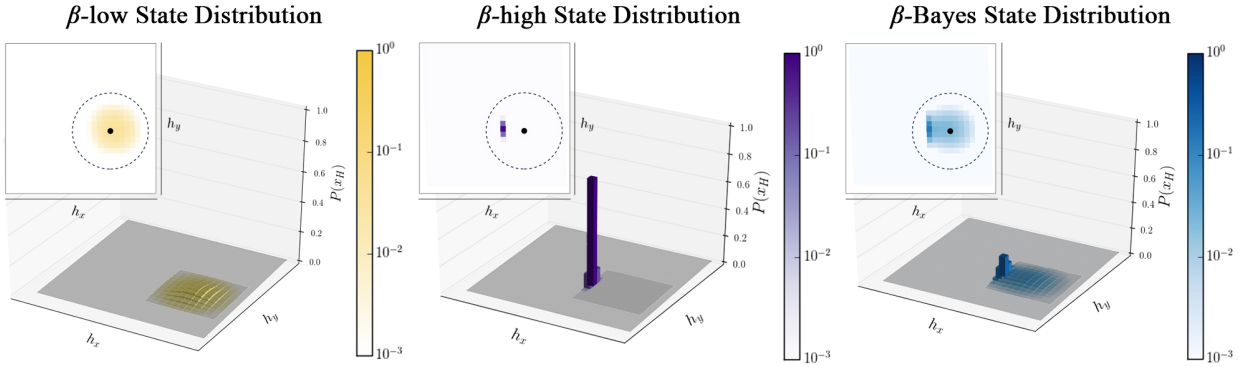


Figure 7.5: Comparison between the predicted human state distribution and the forward-reachable set. The human pedestrian (black dot) is moving in the negative x direction towards a modeled goal. Visualized are the predicted state distributions for 1 second into the future when using low, high, and Bayesian model confidence. Higher-saturation indicates higher likelihood of occupancy. The dashed circle represents the pedestrian's 1-second forward-reachable set.

Given this basic relation, a desirable property for the confidence-aware planning scheme would be that, as confidence in the predictive human behavior model degrades, the high-confidence safety constraints approach, in some meaningful sense, the forward-reachable set avoidance criterion. Indeed, we can see that this is the case: while high-confidence human state predictions will tend to concentrate around near-optimal trajectories according to the utility model encoded by Q_H, low-confidence predictions will spread probability mass more evenly throughout the forward-reachable set. Therefore, when planning its motion to maintain low collision probabilities, the robot will need to keep its robust tracking set $\tilde{\mathcal{X}}(x_R^t)$ clear of a larger portion of the human's forward-reachable set $\mathcal{R}_F(x_H^t;\tau)$ when its confidence in its human behavior predictions is low.

For any grid discretization used to represent the human's state probabilities, we can always find a sufficiently low P_{th} such that, as $\beta \to 0$, the set of states that the robot must avoid equals $\mathcal{R}_F(x_H^t;\tau)$ (up to discretization error): it suffices to choose P_{th} lower than the smallest nonzero probability $P(x_H^\tau)$ in any grid cell for $\beta = 0$. On the other hand, if P_{th} is made much lower than this value, then it may be that, even for some high confidence $\beta > 0$, the very low (yet always nonzero) probabilities in the lowest-utility regions of the forward-reachable set will still be above the threshold, resulting in the entirety of $\mathcal{R}_F(x_H^t;\tau)$ being

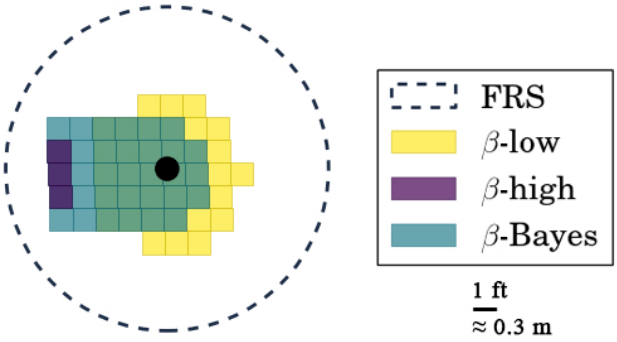


Figure 7.6: Visualization of thresholded human state distributions for different model confidence values. The plot shows discretized states with probability greater than or equal to the collision threshold $P_{\text{th}} = 0.01$, for the probability distributions resulting from fixed low confidence ($\beta = 0.05$) and high confidence ($\beta = 10$), as well as for the posterior marginal state distribution resulting from the Bayesian β inference scheme. By definition, the human's forward-reachable set always includes the set of states assigned probability greater than 0.

avoided. If chosen in the appropriate range, then, the threshold P_{th} can lead to robot planning that mimics worst-case forward-reachable set avoidance only when model confidence decays below a certain level, taking a less conservative approach when the predictive model appears to be accurately capturing the human's behavior.

Figure 7.6 depicts the set of states with predicted probability mass greater than $P_{\text{th}} = 0.01$ overlaid on the human's forward reachable set at time τ, which is a circle of radius $v_H(\tau - t)$ centered on x_H^t for the dynamics in the running example. When model confidence is high ($\beta = 10$), we observe that virtually all of the probability mass is concentrated in a small number of states in the direction of motion predicted by the utility model. When model confidence is low ($\beta = 0.05$) we observe that the set of states assigned probability above the collision threshold P_{th} occupies a much larger fraction of the reachable set. A typical belief $b(\beta)$ recorded at a moment when the human was roughly moving according to Q_H yields an intermediate set of states.

Therefore, we see that the confidence-aware prediction method generally trades lower conservativeness for a probabilistic relaxation of the robust collision avoidance guarantees. The probability threshold P_{th} allows system designers to regulate the degree of conservativeness in the resulting robot motion plan, which can be made to automatically approach the more conservative forward-reachable set avoidance criterion when confidence in the predictive model degrades severely.

7.4 Demonstration with Real Human Trajectories

We implemented real-time human motion prediction with β inference and safe probabilistic motion planning via FaSTrack within the Robot Operating System (ROS) framework [142]. To demonstrate the characteristic behavior of our approach, we created three different environment setups and collected a total of 48 human walking trajectories (walked by 16 different people). The trajectories are measured as (x, y) positions on the ground plane at roughly 235 Hz by an OptiTrack infrared motion capture system.[5] We also demonstrated our system in hardware on a Crazyflie 2.0 platform navigating around a person in a physical space.

Environments. In the first environment there are no obstacles and the robot is aware of the human's goal. The second environment is identical to the first, except that the human must avoid a coffee spill that the robot is unaware of. In the third environment, the human walks in a triangular pattern from her start position to two known goals and back.

Evaluated Methods. For each human trajectory, we compare the performance of our adaptive β inference method with two baselines using fixed $\beta \in \{0.05, 10\}$. When $\beta = 0.05$, the robot is unsure of its model of the human's motion. This low-confidence method cannot trust its own predictions about the human's future trajectory. On the other hand, the $\beta = 10$ high-confidence method remains confident in its predictions even when the human deviates from them. These two baselines exist at opposite ends of a spectrum. Comparing our adaptive inference method to these baselines provides useful intuition for the relative performance of all three methods in common failure modes (see Figure 7.4).

Metrics. We measure the performance of our adaptive β inference approach in both of these cases by simulating a quadrotor moving through the environment to a pre-specified goal position while replaying the recorded human trajectory. We simulate near-hover quadrotor dynamics with the FaSTrack optimal controller applied at 100 Hz. For each simulation, we record the minimum distance in the ground plane between the human and the quadrotor as a proxy for the overall safety of the system. The quadrotor's travel time serves to measure its overall efficiency.

In each environment, we compute the safety metric for all 16 human trajectories when applying each of the three human motion prediction methods and display the corresponding box and whisker plots side by side. To compare the efficiency of our approach to the baselines we compute the difference between the trajectory completion time of our approach, T_{infer}, and that of the low and high confidence baselines, $\{T_{\text{lo}}, T_{\text{hi}}\}$. If the resulting boxplots are below zero, then β inference results in faster robot trajectories than the baselines on a per-human trajectory basis.[6]

Complete Model. First, we designed an example environment where the robot's model is complete and the human motion appears to be rational. In this scenario, humans would

[5] We note that in a more realistic setting, we would need to utilize alternative methods for state estimation such as lidar measurements.

[6] The upper and lower bounds of the box in each boxplot are the 75^{th} and 25^{th} percentiles. The horizontal red line is the median, and the notches show the bootstrapped 95% confidence interval for the population mean.

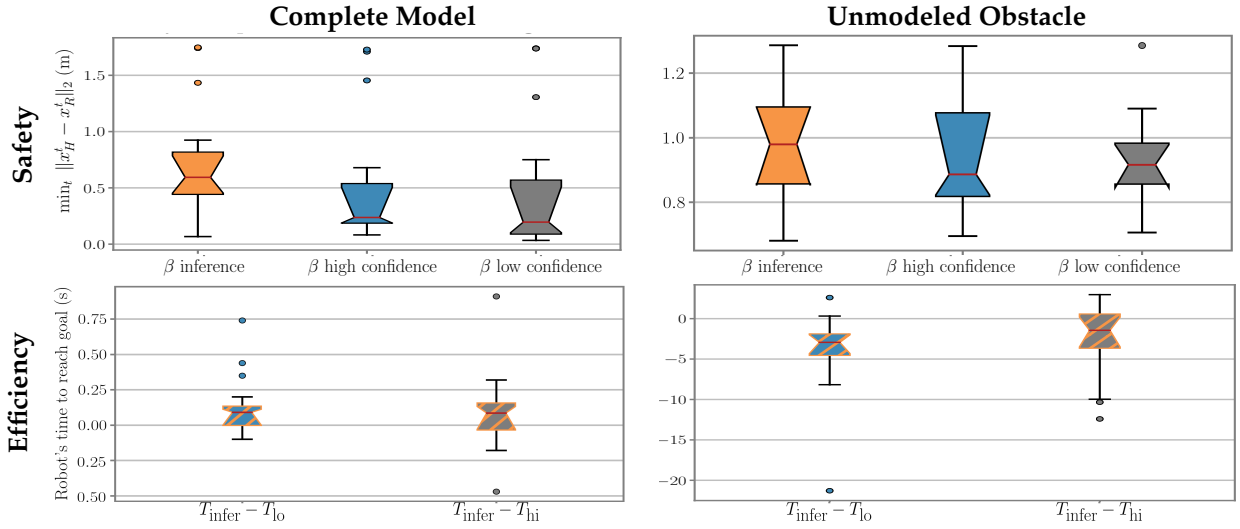


Figure 7.7: Safety and efficiency metrics in a correctly modeled environment and one with an unmodeled obstacle.

walk in a straight line from their start location to their goal which was known by the robot *a priori*.

When the robot has high confidence in its model, the human's direct motion towards the goal appears highly rational and results in both safe (Figure 7.7, top left) and efficient plans (Figure 7.7, bottom left). We see a similar behavior for the robot that adapts its confidence: although initially the robot is uncertain about how well the human's motion matches its model, the direct behavior of the human leads to the robot to believe that it has high model confidence. Thus, the β inference robot produces overall safe and efficient plans. Although we expect that the low-confidence model would lead to less efficient plans but comparably safe plans, we see that the low-confidence robot performs comparably in terms of both safety and efficiency.

Ultimately, this example demonstrates that when the robot's model is rich enough to capture the environment and behavior of the human, inferring model confidence does not hinder the robot from producing safe and efficient plans.

Unmodeled Obstacle. Often, robots do not have fully specified models of the environment. In this scenario, the human has the same start and goal as in the complete model case except that there is a coffee spill in her path. This coffee spill on the ground is unmodeled by the robot, making the human's motion appear less rational.

When the human is navigating around the unmodeled coffee spill, the robot that continuously updates its model confidence and replans with the updated predictions almost always maintains a safe distance (Figure 7.7, top right). In comparison, the fixed-β models that have either high-confidence or low-confidence approach the human more closely. This increase in the minimum distance between the human and the robot during execution time indicates that continuous β inference can lead to safer robot plans.

For the efficiency metric, a robot that uses β inference is able to get to the goal faster than a robot that assumes a high or a low confidence in its human model (Figure 7.7, bottom right). This is particularly interesting as overall we see that enabling the robot to reason about its model confidence can lead to *safer* and *more efficient* plans.

Unmodeled Goal. In most realistic human-robot encounters, even if the robot does have an accurate environment map and observes all obstacles, it is unlikely for it to be aware of all human goals. We test our approach's resilience to unknown human goals by constructing a scenario in which the human moves between both known and unknown goals. The human first moves to two known goal positions, then back to the start. The first two legs of this trajectory are consistent with the robot's model of goal-oriented motion. However, when the human returns to the start, she appears irrational to the robot. Figure 7.8 and 7.9

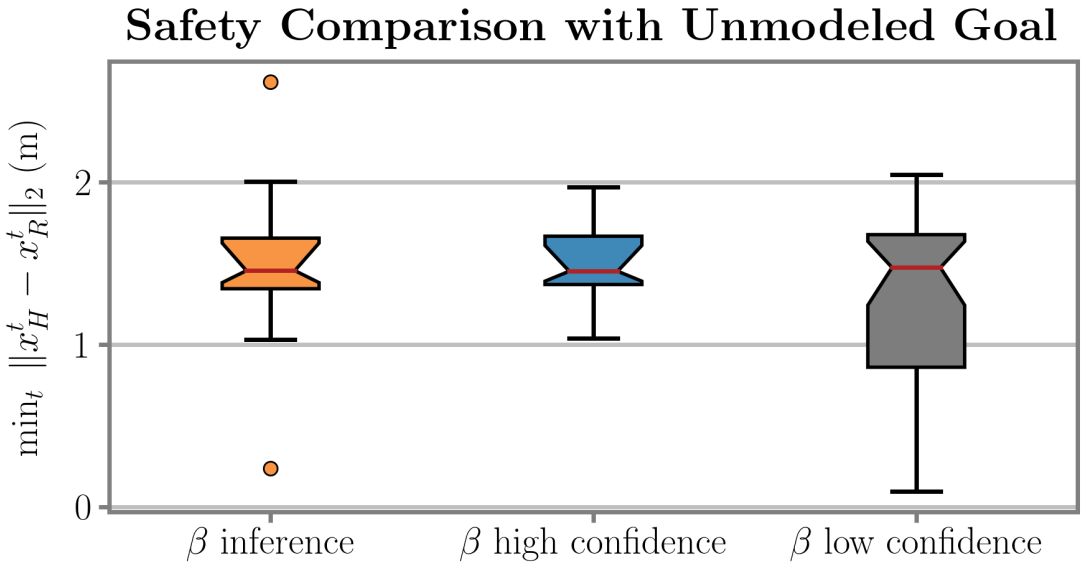


Figure 7.8: Safety results for the unmodeled human goal scenario.

summarize the performance of the inferred-β, high-confidence, and low-confidence methods in this scenario. All three methods perform similarly with respect to the minimum distance safety metric in Figure 7.8. However, Figure 7.9 suggests that the inferred-β method is several seconds faster than both fixed-β approaches. This indicates that, without sacrificing safety, our inferred-β approach allows the safe motion planner to find more efficient robot trajectories.

7.5 Safe Multi-Human Multi-Robot Navigation

We now consider the use of confidence-aware predictions for trajectory planning in the midst of multiple humans, as well as in coordination among multiple robots. A representative

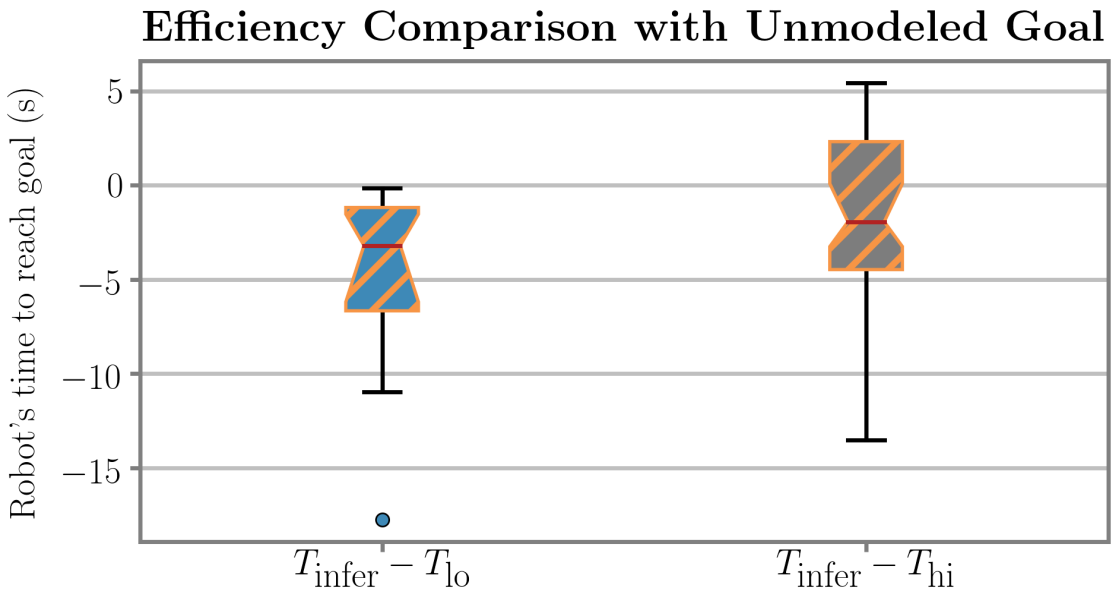


Figure 7.9: Efficiency results for the unmodeled human goal scenario.

hardware demonstration is depicted in Figure 7.10. As we saw in Chapter 4, providing safety assurances for complex systems with multiple agents is an important technical problem, often one with intrinsic scalability challenges due to combinatorial explosion and the *curse of dimensionality*.

Two main difficulties emerge when considering the multi-human prediction problem. On the one hand, there is a fundamental modeling challenge: while low-dimensional state and utility representations (such as the ones used in this chapter) have been seen to yield reasonably accurate predictions for a single human [79, 178], this observation does not easily translate to scenarios including relevant multi-human interactions. For example, two humans currently walking towards each other may sidestep to avoid colliding and continue on their paths, but they may also stop to greet or converse with each other; or they may come into an embrace, effectively colliding. Forming accurate predictions in such scenarios therefore tends to require considerably richer and more subtle model representations.

On the other hand, there is a computational challenge: even if a sufficiently accurate human interaction model is available, it is then necessary to use this model to generate predictions about the joint future actions of multiple humans. The existence of interactions implies that predictions for different humans will in general be coupled with each other. Reasoning over possible joint trajectories therefore carries a combinatorial computational cost, making real-time implementation a difficult problem.

There is a wealth of literature in multi-human prediction, with methods that seek to balance the two challenges in different ways.[7] Simple characterizations of interaction such

[7] We direct the interested reader to [190] for a comprehensive overview of multi-human prediction in the

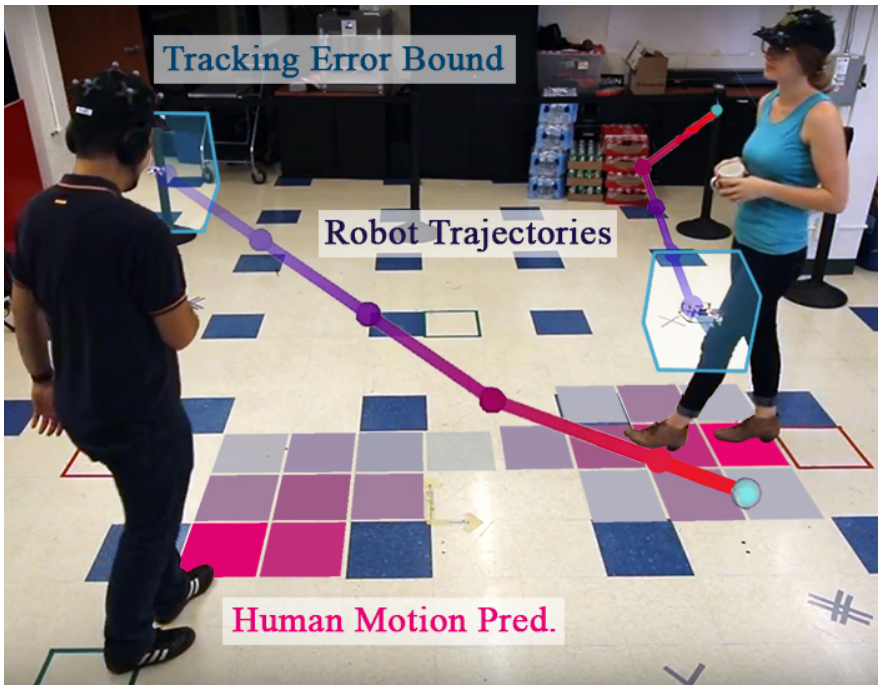


Figure 7.10: Hardware demonstration of real-time multi-agent planning while maintaining safety with respect to internal dynamics, external disturbances, and humans. The quadrotor's trajectories are visualized, and the robust tracking set is shown as a box around each quadrotor. The predictions of future human motion is shown in pink in front of each human. Video: `https://youtu.be/lJGRHNJ1_Wk`

as *social force* models [191] encode explicit assumptions about interaction dynamics in the interest of forming tractable simulation-based predictions. Models of this class are commonly used due to real-time tractability, yet their predictive accuracy degrades substantially when the underlying assumptions are not met. More recent work has focused on incorporating semantic scene information into the prediction of more complex human behavior; some of these models focus on individual human agents, explicitly neglecting interactions [192], whereas others seek use the semantic structure to inform interaction predictions [193].

Whatever the model used, the inevitable occurrence of unmodeled or poorly modeled interactions exacerbates the difficulty in predicting human motion, while computational limits tend to impose decoupling simplifications when forming real-time predictions. Such simplifications, made at computation time, can induce additional inaccuracies. The confidence-aware framework can be applied to multi-human predictions by adjusting the uncertainty around the motion of each human based on the observed accuracy of the chosen model in describing her motion: when the interaction model becomes inaccurate, prediction confidence will automatically be lowered, leading to more conservative planning as with the

specific context of pedestrian motion.

single-human case. We observe that this automatic uncertainty adjustment allows us to simplify and even neglect interaction effects between human individuals in the predictive models while retaining safe and desirable robot behavior.

We will next discuss the reasoning of a single robot about the probability of future collisions with a set of N_H humans. Aggregating such analysis to compute coordinated trajectories for a set of N_R robots is then straightforward under the STP scheme.

7.5.1 Multi-Human Prediction Formulation

Let the dynamics of each human $i = 1, \ldots, N_H$ be generally given by

$$\dot{x}_i = f_i(x_i, u_i) \ . \tag{7.19}$$

In addition let x denote the joint state of all agents (humans and robots) for planning purposes; we assume x is observable to all agents.

As before, the robot needs to plan a trajectory that, when tracked by the physical system, will reach a goal state as efficiently as possible while avoiding, with high probability, the joint danger zones $\mathcal{Z}_{i,R}$ with respect to all humans i, based on an informed prediction of their future motion (in addition to staying clear of static obstacles and other robots). Following the analysis for the single-human case, we can write the robot's probabilistic safety condition (7.7e) as:

$$P_{\text{coll}}^{t:T} := P\big(\exists \tau \in \{t, ..., T\} : x_1^\tau \in \mathcal{H}_1(x_R^\tau) \vee \ldots \vee x_{N_H}^\tau \in \mathcal{H}_{N_H}(x_R^\tau)\big) \leq P_{\text{th}} \tag{7.20}$$

with each $\mathcal{H}_i$ defined analogously to (7.14).

The robot may model each human through a noisily rational probabilistic policy similar to (7.6). However, in the general case, the human's objective (and therefore her state-action value function) may now depend on the overall state of the system x^τ, including all other agents (human and robotic), and not only her own individual state x_i^τ.

$$P(u_i^t \mid x^t; \beta_i, \theta_i) \propto e^{\beta_i Q_i(x^t, u_i^t; \theta_i)} \ . \tag{7.21}$$

Depending on the interaction model used, determining this Q-value for each human is a nontrivial problem. For example, if the humans are assumed to behave in a strategic way accounting for each other's future decisions, the full solution to $\{Q_i\}_{i=1}^N$ is game-theoretic and, if at all tractable, may need to be numerically precomputed offline [41]. The choice of an appropriate interaction model that will allow tractable but informative computation of player values and strategies is an open area of research that we will explore further in Chapter 8. Our interest here, however, is not in the choice of a certain interaction model, but rather the ability to mitigate the impact of model inaccuracies through confidence-aware predictions.

Given a state-action value function for each human, we have a well-defined probabilistic policy model for their behavior. As in the single-human case, generating informed probabilistic predictions at each instant will require keeping an updated Bayesian belief over

the appropriate model confidence β_i for each human based on the observed behavior. This Bayesian update step, under the confidence-aware prediction framework introduced in this chapter, scales remarkably well with the number of humans N_H. In fact, the Bayesian update (7.9) can be performed independently for each i-th human given the observed state x, which means that the total amount of computation is linear in N_H and, if parallel computation is available, computation *time* can be made constant in N_H.

With this in place, the last central step is to compute the probability distribution over the N_H humans' joint state at each future time step τ, which is needed in order to evaluate (7.20). These operations can become extremely computationally intensive when explicitly using a joint probability distribution, which would in general have to be discretized and represented numerically as a grid of $n_1 + \ldots + n_{N_H}$ dimensions. Operating with (or even storing) such a grid becomes rapidly impractical as N_H grows beyond trivial numbers. Instead, tractable approximations can be computed by only storing the marginal predicted distribution of each human at every future time step τ. This way, the robot need only operate with N_H grids, each of n_i dimensions. Treating humans' state distributions as independent of each other at each instant leads to a computationally simple noisy-OR operation for the marginal collision probability at each future τ:

$$P_{\text{coll}}^{\tau} \approx 1 - \prod_{i=1}^{N_H} \Big(1 - P\big(x_i^{\tau} \in \mathcal{H}_i(x_R^{\tau})\big)\Big) \ . \tag{7.22}$$

The approximation in (7.22) becomes exact for models in which the probabilities are in fact independent, and in particular for any of the large class of human models in the literature that altogether neglect interaction.

7.5.2 Hardware Demonstration

In the following demonstration, we consider a simple model where human pedestrians follow the dynamics in (7.3). We use a simple human motion model that assumes *no interaction*, and let the confidence-aware scheme treat any relevant interaction effects as inconsistencies between the model and reality, appropriately lowering prediction confidence.

We implemented our framework in C++ and Python, using Robot Operating System (ROS) [142]. As shown in Figure 7.10, we used Crazyflie 2.0 quadrotors as our robots, and two human volunteers. All computations for our hardware demonstration were done on a single laptop computer (specs: 31.3 GB of memory, 216.4 GB disk, Intel Core i7 @ 2.70GHz x 8), due to the limited onboard computational capability of the Crazyflie robots.[8] The position and orientation of robots and humans were measured at roughly 235 Hz by an OptiTrack infrared motion capture system. The humans were instructed to move towards different places in the lab, while the quadrotors planned collision-free trajectories in three dimensions (x, y, z) using a space-time implementation of A*. The quadrotors tracked these trajectories

[8]Despite running on a single computer, our implementation is decentralized within the ROS ecosystem.

using the precomputed FaSTrack controller designed for a 6D near-hover quadrotor model tracking a 3D point [16]. This corresponds to a robust STP scheme (as introduced in Chapter 4) replacing the Hamilton-Jacobi-based minimum-time planning in Steps **R**–**X** of Algorithm 4.2 with space-time A* search. Human motion was predicted 2 s into the future. Figure 7.11 shows several snapshots of this scene over time. The scenario layout requires the humans to move around each other to reach their goals, causing an unmodeled interaction effect. The predictions become less certain during this interaction, and the quadrotors plan more conservatively, giving the humans a wider berth.

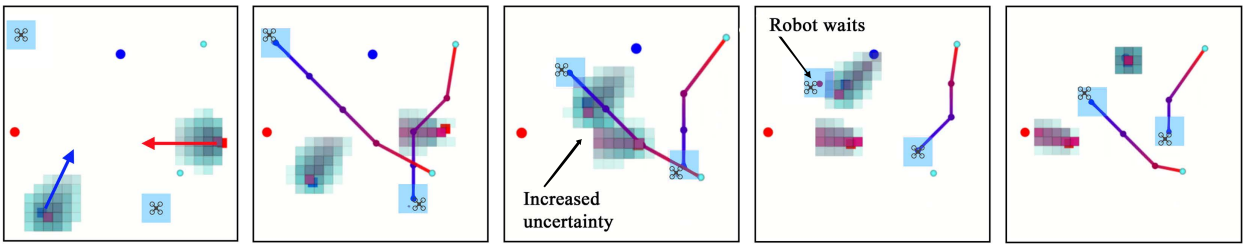


Figure 7.11: Trajectory and prediction visualization of multi-human, multi-robot hardware demonstration. (a) Two humans (red and blue) start moving towards their respective goals (also red and blue). Robot in lower right-hand corner has first priority, and robot in upper left-hand corner has second. The time-varying predictions of each human's future motion are visualized. (b) Robots plan trajectories to their goals based on the predictions, priority order, and are guaranteed to stay within the robust tracking set (shown in blue). (c) When the humans begin to interact in an unmodeled way by moving around each other, the future predictions become more uncertain. (d) The robots adjust their plans to be more conservative–note the upper-left robot waiting as the blue human moves past. (c) When the humans pass each other and the uncertainty decreases, the robots complete their trajectories.

7.6 Implications on Human Preference Inference and Value Alignment

The inability to perfectly model human behavior and reason about human intentions has wider implications beyond the operation of robots *around* humans. Many of the automation systems we build are intended to assist people in a variety of settings, and as such need to make inferences about what people's goals and preferences are. From current intelligent assistants on smartphones to future robotic caregivers, the success of these systems hinges not only on their effectiveness at accomplishing any given task, but on their ability to correctly determine what task needs to be accomplished in order to help their users. Insofar as human behavior and the structure of human preferences are impossible to capture with full accuracy, all of these systems—especially when taking on high-stakes or safety-critical tasks—will need

to competently reason about their own uncertainty as to what their users want. Further, as in the case of the unmodeled goal in Figure 7.3, these systems must acknowledge the possibility that the correct answer may lie outside of their hypothesis space.

Competently reasoning about human preferences is likely to become a major challenge as the capabilities of automation and artificial intelligence technologies continue to increase. This problem was already noted by Norbert Wiener as early as 1960 [194]:

> If we use, to achieve our purposes, a mechanical agency with whose operation we cannot efficiently interfere once we have started it, because the action is so fast and irrevocable that we have not the data to intervene before the action is complete, then we had better be quite sure that the purpose put into the machine is the purpose which we really desire and not merely a colorful imitation of it.

The work in this chapter therefore has deep connections with the problem of *value alignment* in intelligent systems, that is, how to ensure that a system pursues objectives that are truly those of the relevant stakeholders. While traditional formulations across artificial intelligence, control theory, machine learning, and operations research have assumed that a system can be directly given a "ground-truth" objective function to optimize, resulting in desirable behavior, there is growing evidence that designing such an objective function to reliably encode the desired behavior of the system in a wide range of conditions is extremely challenging if not infeasible [195]. This has sparked renewed interest in formulations based on inverse optimal control, seeking to design intelligent systems that actively infer human preferences rather than assuming full knowledge of them from the start [196, 197].

As we have argued in this chapter, while often preferable to assuming a fixed "true" objective, performing inference over a fixed hypothesis space is not necessarily enough either. When no hypothesis in the predefined space accurately captures the observed human behavior, the system is likely to make inappropriate inferences that ultimately result in undesirable behavior.

In a recent study published in [198], we asked human participants to provide demonstrations and real-time corrections to a robot's motion through physical interaction. We observed that when the robot ran inverse optimal control on a fixed hypothesis space it would often reach confident yet incorrect inferences about participants' preferences, which were not always consistent with the assumed model. Instead, the robot could make *confidence-aware* inferences, in an analogous fashion to (7.9), by simultaneously reasoning about the preference parameters (θ) and the confidence parameter (β). The latter approach was successful at mitigating spurious inferences by quantifying the degree to which the different human inputs could be explained within the robot's hypothesis space. However, the question remains as to what the best course of action should be for the robot when it loses confidence in its ability to represent the human's preferences. Should it default to non-committal behavior? Actively ask the human for guidance? Attempt to bootstrap an augmented hypothesis space? How context-specific is the answer, and should it be hard-coded as a design choice or emerge automatically from first-principles decision-making?

Even as we continue to improve the ability of robotic and intelligent systems to accurately model human preferences in the coming years and decades, much of the success of the resulting technologies may hinge on scientific and technical progress on enabling these systems to quantify and respond to the *reality gap* between their mathematical representations and the unobservable internal states of the people they interact with. Ultimately, few technologies can be as potentially catastrophic as a powerful automation system that erroneously assumes complete knowledge of the goal it is supposed to pursue.

7.7 Chapter Summary

This chapter revisits the problem of safety assurance under imperfect characterization of reality, focusing on the important case of human behavior. While impossible to model with full accuracy, human actions tend to be purpose-driven and as such present a great deal of structure, which has been richly studied in the past few decades by cognitive science, econometrics, and related fields. We would therefore like to leverage this structural understanding to inform the decision-making of robotic systems coexisting with human agents, while remaining vigilant of the inevitable limitations of any cognitive model when predicting complex human behavior.

In particular, the work introduced in this chapter builds on the well-established inverse optimal control formulations based on the Boltzmann or *maximum entropy* model of noisily-rational human decision-making. The inverse temperature parameter in these models regulates the entropy of the action distribution for any given intent, and therefore quantifies how confidently a human individual is predicted to choose actions that yield high utility towards said intent. Rather than keeping this parameter fixed during inference, as has been traditionally done, the framework introduced here treats it as a hidden dynamic state encoding the ability of the underlying utility model to accurately capture the ongoing human behavior. By maintaining a real-time Bayesian belief on this *model confidence* parameter, the robot can quickly adapt its forecasts to effectively reflect the predictability of the human's motion in real time.

The resulting *confidence-aware* predictions are then combined with the robust trajectory tracking guarantees introduced in Chapter 5. This requires extending the theoretical analysis the FaSTrack motion planning scheme to handle dynamic probabilistic obstacles such as the ones induced by the uncertain human motion prediction. The result is a novel probabilistic safety certificate that combines worst-case and probabilistic analysis, leading to nominal trajectory plans that robustly keep the physical system clear of collisions with an arbitrarily high probability. We further observe that as this probability threshold approaches 1, trajectory plans avoid the entire *forward-reachable set* of the human (that is, all states that the human could physically be in at a certain future time). For any probability threshold less than 1, the role of confidence-aware prediction can be seen as determining what if any regions of the forward-reachable set the human is unlikely enough to visit that trajectory plans can traverse them with acceptable risk.

Using both recorded human motion and live demonstrations with lightweight quadrotors navigating around human experimenters, we compare the performance of the proposed Bayesian confidence-aware technique to two fixed-confidence approaches, all used in conjunction with the proposed probabilistically safe motion planning scheme. Our results indicate that, even though the three methods perform similarly when the human's motion is well-explained by the robot's model, Bayesian confidence yields safer and more efficient robot trajectories whenever the human's behavior is determined by obstacles or goals that are not part of the robot's model.

We additionally consider scenarios with multiple humans or multiple robots, and seek to exploit structure in the joint planning and prediction space to avoid a combinatorial explosion as the number of agents increases. While the Sequential Trajectory Planning scheme introduced in Chapter 4 allows us to impose this structure among the controlled robotic platforms through a *prescriptive* priority-based planning approach, we cannot make use of such techniques when comes to human individuals. Instead, we use a *descriptive* approach, making structural assumptions about future human trajectories and then allowing the confidence-aware prediction framework to automatically increase uncertainty about the imminent motion of any humans whose actions are not currently well described by the assumed structure. The scheme is demonstrated on a simple scenario with two human pedestrians and two quadrotors: the quadrotors use a simplified predictive model assuming independent human motions, yet successfully maintain safety by increasing separation when the two humans interact substantially to avoid each other.

Something that we have not addressed in this chapter is the complex coupling that may arise between the decisions of robotic systems and human agents. This coupling can play a determining role in interaction-rich settings, such as, for example, in the context of autonomous driving on public roads. We will study these interactions, and the important class of problems they give rise to, in the upcoming Chapter 8. Conversely, whenever robots are executing tasks that require them to remain clear of any humans in their space—such as a robot cleaner vacuuming a room, or a recreational autonomous drone filming its user—it is appropriate to assume that humans will for the most part remain oblivious or indifferent to the presence of the robot. Thus, the robot can follow a "pipeline" approach: first, make predictions about the human's actions as being independent of its own future plans; next, incorporate these predictions into the computation of such plans. Any adjustments that humans might occasionally make in practice to accommodate the robot's motion will simply tend to reduce the amount of maneuvering required on its part. While the independence assumption does mean that the robot will not be counting on such helpful behavior ahead of time, it can nonetheless take advantage of it, through replanning, once it takes place. In cases where interaction progresses in a potentially dangerous way—say, if the human is maliciously attempting to get in the robot's way—the confidence-aware machinery will still recognize this as a departure from model-consistent behavior, automatically increasing the robot's conservativeness and leading it to increase separation.[9]

[9] In fact, the "coffee spill" example depicted in Figure 7.4 could well have been motivated by the human

In the larger context of human-aware robotic motion planning, the confidence-aware prediction framework should be viewed as complementary to the choice of a human model, and not as an alternative choice in itself. In general, it will be desirable to use the most accurate real-time prediction method available: the smaller the discrepancies between the model's predictions and the observed behavior, the smaller the drop in confidence and therefore the less conservative the resulting robot behavior. On the other hand, given that every human model will make some inaccurate predictions, detecting these promptly and increasing conservativeness accordingly is a much preferable alternative to accidentally violating separation constraints due to overconfidence.

Finally, beyond the avoidance of physical collisions between robots and humans, confidence-aware analysis of human behavior can offer useful insights in the problem of preference inference and value alignment, for broader automation and artificial intelligence systems. As these systems become increasingly capable, it becomes paramount that they competently reason about the needs and intentions of human stakeholders, as well as their own ability to accurately model them.

actively attempting to intercept the quadrotor rather than avoiding a spill on the floor: the confidence-aware scheme is by definition agnostic to the causes behind unmodeled human behavior.

Chapter 8

Game-Theoretic Autonomous Driving

> Patience is something you
> admire in the driver behind you
> and scorn in the one ahead.
>
> Mac McCleary

This chapter is based on the paper "Hierarchical Game-Theoretic Planning for Autonomous Vehicles" [21], written in collaboration with Eli Bronstein, Elis Stefansson, Dorsa Sadigh, Shankar Sastry, and Anca Dragan.

Reasoning about safe robotic operation in spaces shared with human beings poses a major challenge as these technologies become widely deployed. In Chapter 7, we looked into how robots may reason at runtime about the accuracy and reliability of their predictive models of human behavior. However, we explicitly eschewed the treatment of coupled interactions between robots and humans, and instead assumed that human decisions were approximately indifferent to the presence of robots. While this assumption is a useful one in certain applications, there exist important settings in which human behavior cannot be accurately explained (or predicted) without explicitly accounting for interaction—not only to the extent that robot actions may influence human decisions, but also due to the expectations that humans themselves may have about how their own actions will affect the decisions of robotic systems around them. This coupling between the decisions of the autonomous system and those of other agents transcends the "unilateral" formulations in classical robotic motion planning and optimal control theory, and necessarily places us in the realm of dynamic game theory.

Perhaps no modern robotic application highlights the criticality of strategic interaction as clearly as autonomous driving: the actions of a self-driving vehicle on the road permanently affect and are affected by those of other drivers, whether overtaking, negotiating a merge, or avoiding an accident. This creates a strong coupling between the vehicle's planning

and its predictions of other drivers' behavior, and constitutes an open problem with direct implications on the safety and viability of autonomous driving technology.

Dynamic game formulations have by and large been deemed too computationally demanding to meet the real-time constraints of autonomous driving in its continuous state and action space. In this chapter, we introduce a novel game-theoretic trajectory planning algorithm for autonomous driving, that enables real-time performance by hierarchically decomposing the underlying dynamic game into a long-horizon "strategic" game with simplified dynamics and full information structure, and a short-horizon "tactical" game with full dynamics and a simplified information structure. The value of the strategic game is used to guide the tactical planning, implicitly extending the planning horizon, pushing the local trajectory optimization closer to global solutions, and, most importantly, quantitatively accounting for the autonomous vehicle and the human driver's ability and incentives to influence each other. In addition, our approach admits non-deterministic models of human decision-making, rather than relying on perfectly rational predictions. Our results showcase richer, safer, and more effective autonomous behavior in comparison to existing techniques.

The Promise and Challenge of Autonomous Driving

Imagine you are driving your car on the highway and, just as you are about to pass a large truck on the other lane, you spot another car quickly approaching in the wing mirror. Your driver's gut immediately gets the picture: the other driver is trying to squeeze past and cut in front of you at the very last second, barely missing the truck. Your mind races forward to produce an alarming conclusion: it is too tight—yet the other driver seems determined to attempt the risky maneuver anyway. If you brake immediately, you could give the other car enough room to complete the maneuver without risking an accident; if you accelerate, you might close the gap fast enough to dissuade the other driver altogether. What do you decide?

Driving is fundamentally a game-theoretic problem, in which road users' decisions continually couple with each other over time. Gracefully negotiating these continual interactions is central to preserving safety on the road, a task that human beings are—for the most part—highly successful at. In 2018, the rate of motor vehicle crash deaths in the United States was 1.13 per 100 million miles traveled [199], a remarkably small number considering the wide range of visibility, weather, and traffic conditions aggregated by this statistic. Unfortunately, while the rate may seem small, absolute numbers paint a more somber picture: 2018 closed with a computed death toll of 36,560 in the United States alone, with an estimated worldwide total of 1.35 million a year—or 3700 deaths every day—having become the leading cause of death for people between 5 and 29 years of age [200]. In studies conducted by the U.S. National Highway Traffic Safety Administration (NHTSA) 94% of car crashes had driver error as the critical cause [201].

Autonomous driving technology therefore holds an immense potential to drastically reduce road fatalities, yet in order to do this, it must reliably surpass, or at the very least match,

the high safety competence of unimpaired human drivers, something that will inescapably require accurately planning through road interactions.

Related Work

Most approaches in the literature follow a "pipeline" approach that generates predictions of the trajectories of human-driven vehicles and then feeds them to the planning module as unalterable moving obstacles [202–205]. This can lead to both excessively conservative and in some cases unsafe behavior, a well-studied issue in the robotic navigation literature known as the "frozen robot" problem [206]. Indeed, some of the prototype self-driving vehicles undergoing testing on public roads in recent years have been repeatedly reported to struggle to merge onto highways even under moderate traffic conditions [207, 208].

Recent work has addressed this by modeling human drivers as utility-driven agents who will plan their trajectory in response to the autonomous vehicle's internal plan. The autonomous vehicle can then select a plan that will elicit the best human trajectory in response [209, 210]. Unfortunately, this treats the human as a pure *follower* in the game-theoretic sense, effectively inverting the roles in previous approaches. That is, the human is assumed to take the autonomous vehicle's future trajectory as immutable and plan her own fully accommodating to it, rather than try to influence it. Further, the human driver must be able to observe, or exactly predict, the future trajectory planned by the autonomous vehicle, which is unrealistic beyond very short planning horizons.

Ultimately, incorrectly predicting the behavior of human road users can lead to tragic consequences when deploying autonomous vehicles—or even when testing them. In 2018, a prototype self-driving vehicle struck a pedestrian who was crossing a street outside a crosswalk in Tempe, Arizona; after a lengthy investigation, the U.S. National Transportation Safety Board (NTSB) recently published an accident report in which the automated driving system's consistent failure to correctly predict the pedestrian's motion is listed as one of the causes that led to the collision [211].[1]

Contribution

In this work, we introduce a hierarchical game-theoretic framework to address the mutual influence between the human and the autonomous vehicle while maintaining computational tractability. In contrast with recent game-theoretic planning schemes that assume open-loop information structures [212–214], the proposed framework hinges on the use of a fully coupled interaction model in order to plan for horizons of multiple seconds, during which drivers can affect each other's behavior through their actions over time. We do this by computing the

[1]However, the determining cause identified by the report was the human safety operator's failure to monitor the vehicle and its environment because "she was visually distracted throughout the trip by her personal cell phone". Automation complacency and overreliance are serious concerns in human-automation systems, and while we do not address them here directly, they are likely to become increasingly critical as the complexity and penetration of robotic technologies continues to progress.

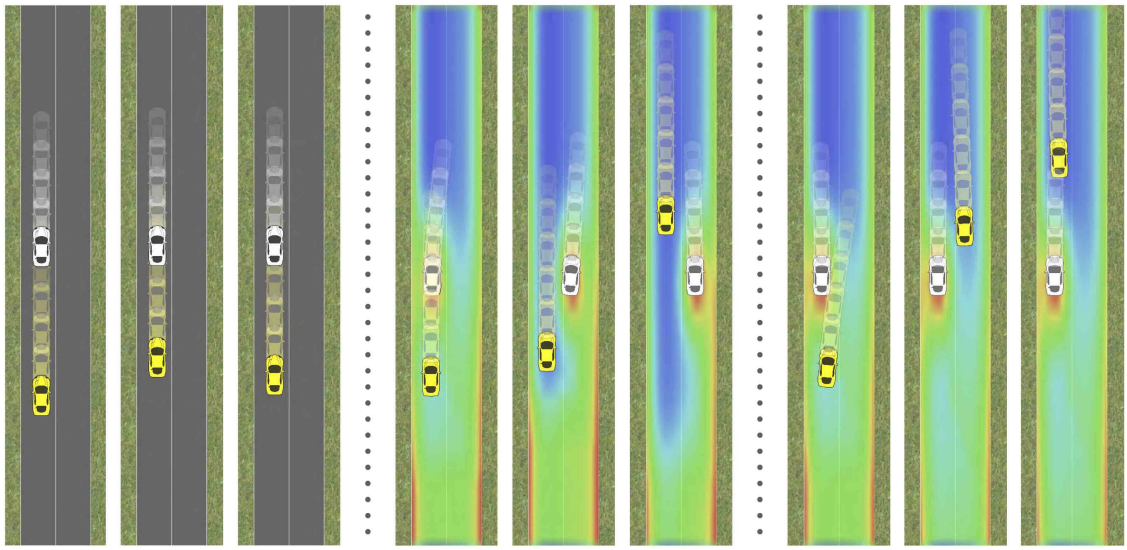


Figure 8.1: Demonstration of the proposed hierarchical game-theoretic planning framework on a simulated overtaking scenario. The heatmap displays the hierarchical planner's strategic value, ranging from red (low value) to blue (high value), which accounts for the outcome of possible interactions between the two vehicles. *Left:* Using a short-horizon trajectory planner, the autonomous vehicle slows down and is unable to overtake the human. *Center:* Using the hierarchical game-theoretic planner, the autonomous vehicle approaches the human from behind, incentivizing her to change lanes and let it pass (note the growth of a high-value region directly behind the human in the left lane). *Right:* If the human does not maneuver, the autonomous vehicle executes a lane change and overtakes, following the higher values in the right lane.

optimal value and strategies for a dynamic nonzero-sum game with a long horizon (typically a few seconds) and a full closed-loop feedback information structure [42, 43]. In order to maintain tractability, we propose solving this long-horizon game using simplified dynamics, which will approximately capture the vehicles' ability to execute different trajectories. The resulting long-term value, which captures the expected outcome of the strategic interaction from every state, can then be used as an informative terminal component in the objective function used in a receding-horizon planning and control scheme. This low-level planner can use a higher-fidelity representation of the dynamics, while only planning for a shorter time horizon (typically less than one second) during which simplifications in the interaction have a less critical effect; this is supported by a substantial body of work in the human-factors engineering literature, where *preview* models, based on simple predictive control schemes, have been observed to describe human driver responses over short horizons (typically in the order of one second) [215–217].

The proposed framework therefore hierarchically combines:

- A *strategic* (high-level) planner that determines the outcome of long-term interactions

using simplified dynamics and fully coupled interaction.

- A *tactical* (low-level) planner that computes short-term vehicle trajectories using high-fidelity dynamics and simplified interaction, informed by the long-term value computed by the strategic planner.

Thanks to the more accurate interaction model and the more tractable dynamical model, the hierarchical framework makes it possible to reason farther into the future than most receding-horizon trajectory planners. The high-level game value informs the trajectory optimization as a terminal cost, implicitly giving it an approximate insight into the longer time scale (in a similar spirit to a variety of planning schemes, e.g. [8]). In addition, since this strategic value is computed globally via dynamic programming, it can help mitigate the local nature of most trajectory optimization schemes, biasing them towards better solutions.

An important strength of the proposed framework is that the strategic planner does not require using a deterministic model of the human, such as an ideal rational agent, but instead allows a variety of models including probabilistic models such as noisy rationality, commonly used in inverse optimal control (also inverse reinforcement learning) [79, 189]. In addition, the framework is agnostic to the concrete planner used at the tactical level: while this chapter demonstrates the approach with a trajectory optimizer based on [209], this could be replaced with other methods, including deep closed-loop prediction models, such as [186], by introducing the strategic value as a terminal cost term in their objective function. Therefore, the method proposed here should not be seen as competing with such planning schemes, but rather as complementing them.

Importantly, solving the underlying dynamic game does not imply that the autonomous vehicle will be more selfish or aggressive—its driving behavior will ultimately depend on the optimization objective specified by the system designer, which may include terms encoding comfort and safety of other road users. With adequate objective design, the proposed framework can enable safer and more efficient autonomous driving by planning with a more accurate model of interactions.

The formulation presented here can in principle be extended to N players and equilibrium solutions are well-defined in *theory* given an information structure, in practice solving the full multiplayer game requires exponential computation in the number of interacting vehicles or agents, which constitutes a fundamental open problem. We thus limit the scope of this work to pairwise interactions, and note that tractable extensions may be achieved through decomposition strategies in the lines of those introduced in Chapter 4 and prediction approaches that degrade gracefully with inaccurately modeled interactions, such as those discussed in Chapter 7.

8.1 Driving as a Nonzero-Sum Dynamic Game

We consider a single human driver H and a single autonomous system A in control of their respective vehicles. The dynamics of the joint state $x^t \in \mathbb{R}^n$ of the vehicles in the world,

which we assume to be fully observable, are

$$x^{t+1} = f(x^t, u_A^t, u_H^t) \ , \tag{8.1}$$

where $u_i^t \in \mathcal{U}_i \subset \mathbb{R}^{n_{ui}}$ is the driving control action for each $i \in \{A, H\}$ at time step t; we assume $\mathcal{U}_i$ is compact.

The autonomous system is attempting to maximize an objective that depends on the evolution of the two vehicles over some finite time horizon, namely a cumulative return:

$$R_A(x^{0:N}, u_A^{0:N}, u_H^{0:N}) = \sum_{t=0}^{N} L_A(x^t, u_A^t, u_H^t) \ . \tag{8.2}$$

The reward function L_A captures the designer's specifications of the vehicle's behavior and may encode aspects like fuel consumption, passenger comfort, courteousness, time efficiency, and safety . Some of these aspects (crucially safety) may depend on the joint state of the two vehicles; the reward function may also explicitly depend on the human driver's actions (the designer may, for instance, decide to penalize it for causing other vehicles to maneuver abruptly). The autonomous vehicle therefore needs to reason about not only its own future actions, but also those of the human driver.

We assume that the autonomous vehicle has some predictive model of the human's actions as a function of the currently available information (the joint state, and possibly the autonomous vehicle's current action). The coupling in the planning problem is then explicit. If the system models the human as exactly or approximately attempting to maximize her own objective function, the coupling takes the form of a dynamic game, in which each player acts strategically as per her own objective function accounting for the other's possible actions. Since both players observe the current state at each time, this dynamic game has closed-loop feedback information structure, and optimal values and strategies can be computed using dynamic programming [42, 43].

Unfortunately, deriving these strategies can be computationally prohibitive due to the exponential scaling of computation with the dimensionality of the joint state space (which will be high for the dynamical models used in vehicle trajectory planning). However, we argue that successfully reasoning about traffic interactions over a horizon of a few seconds does not require a full-fidelity model of vehicle dynamics, and that highly informative insights can be tractably obtained through approximate models. We further argue that it is both useful and reasonable to model human drivers as similarly reasoning about vehicle interactions over the next few seconds without needing to account for fully detailed vehicle dynamics. This insight is at the core of our solution approach.

8.2 Hierarchical Game-Theoretic Planning

We propose a hierarchical decomposition of the interaction between the autonomous vehicle and the human driver. At the high level, we solve a dynamic game representing the long-horizon interaction between the two vehicles through approximate dynamics. At the low

level, we use the players' computed value functions as an approximation of the best long-horizon outcome achievable by both vehicles from each joint state, and incorporate it in the form of a guiding terminal term in the short-horizon trajectory optimization, which is solved in a receding-horizon fashion with a high-fidelity model of the vehicles' dynamics.

8.2.1 Strategic planner: Closed-loop dynamic game

Let the approximate dynamics be given by

$$s^{k+1} = \phi(s^k, a_A^k, a_H^k) \ , \tag{8.3}$$

where $s^t \in \mathbb{R}^{\tilde{n}}$ and $a_i^t \in \mathcal{A}_i \subset \mathbb{R}^{\tilde{n}_{u_i}}$ are the state and action in the simplified dynamics ϕ. The index k is associated to a discrete time step that may be equal to the low-level time step or possibly different (typically coarser). We generically assume that there exists a function $g : \mathbb{R}^n \to \mathbb{R}^{\tilde{n}}$ assigning a simplified state $s \in \mathbb{R}^{\tilde{n}}$ to every full state $x \in \mathbb{R}^n$. The approximation is usually made seeking $\tilde{n} < n$ to improve tractability. This can typically be achieved by ignoring dynamic modes in f_i with comparatively small time constants. For example, we may assume that vehicles can achieve any lateral velocity within a bounded range in one time step, and treat it as an input instead of a state.[2]

We model the dynamic game under feedback closed-loop information (both players' actions can depend on the current state s but not on the state history), allowing the human driver to condition her choice of a_H^k on the autonomous vehicle's current action a_A^k at every time step k, resulting in a Stackelberg (or leader-follower) dynamic game [43]. As in Chapter 7, we need not assume that the human is an ideal rational player, but can instead allow her action to be drawn from a probability distribution. Typically, we may have used inverse optimal control methods [79, 218] to learn a set of driver preferences, which would then inform our predictions of future actions. As we saw in Chapter 7, these probabilistic predictions can also be modulated to account for modeling inaccuracies, to the extent that the human driver's behavior will inevitably depart from the modeling assumptions.

We generalize the well-defined feedback Stackelberg dynamic programming solution [42] to the case in which one of the players, in this case the *follower*, has a noisy decision rule (often referred to as a *quantal response*): $p(a_H^k | s^k, a_A^k)$. The autonomous vehicle, here in the role of the *leader*, faces at each time step k the nested optimization problem of selecting the action with the highest state-action Q value, which depends on the human's decision rule p, in turn affected by the human's own Q values:

$$\max_{a_A^k} Q_A^k(s^k, a_A^k) \tag{8.4a}$$

$$\text{s.t. } p(a_H^k \mid s^k, a_A^k) = \pi_H\big[Q_H^k(s^k, a_A^k, \cdot)\big](a_H^k) \tag{8.4b}$$

[2] These simplified dynamics can be seen as analogous to the *planning dynamics* we utilized in Chapter 5; however, rather than focusing on tracking nominal trajectories, here we will use the approximate dynamics to inform the high-fidelity planner by augmenting its objective function.

where Q_A^k and Q_H^k are the state-action value functions at time step k, and $\pi_H : L^\infty \to \Delta(\mathcal{A}_H)$ maps every utility function $q : \mathcal{A}_H \to \mathbb{R}$ to a probability distribution over $\mathcal{A}_H$. A common choice of π_H, as discussed in Chapters 2 and 7, is the Boltzmann-rational policy, for which:

$$P(a_H \mid s, a_A) \propto e^{\beta Q_H(s, a_A, a_H)} \ . \tag{8.5}$$

The values Q_A^k and Q_H^k are recursively obtained in backward time through successive application of the dynamic programming equations for $k = K, K-1, \ldots, 0$:

$$\pi_A^*(s) := \arg\max_a Q_A^{k+1}(s, a) \ , \quad \forall s \in \mathbb{R}^{\tilde{n}} \tag{8.6a}$$

$$a_H^i \sim \pi_H\big[Q_H^i(s^i, a_A^i, \cdot)\big] \ , \quad i \in \{k, k+1\} \tag{8.6b}$$

$$Q_H^k(s^k, a_A^k, a_H^k) = \tilde{L}_H(s^k, a_A^k, a_H^k) + \mathbb{E}_{a_H^{k+1}} Q_H^{k+1}(s^{k+1}, \pi_A^*(s^{k+1}), a_H^{k+1}) \tag{8.6c}$$

$$Q_A^k(s^k, a_A^k) = \mathbb{E}_{a_H^k} \tilde{L}_A(s^k, a_A^k, a_H^k) + Q_A^{k+1}(s^{k+1}, \pi_A^*(s^{k+1})) \tag{8.6d}$$

with s^{k+1} from (8.3) and letting $Q_A^{K+1} \equiv 0$, $Q_H^{K+1} \equiv 0$.

The solution approach is presented in Algorithm 8.1 for a discretized state and action grid $\hat{\mathcal{S}} \times \hat{\mathcal{A}}_A \times \hat{\mathcal{A}}_H$. This computation is typically intensive, with complexity $O\big(|\hat{\mathcal{S}}| \cdot |\hat{\mathcal{A}}_A| \cdot |\hat{\mathcal{A}}_H| \cdot K\big)$, but is also extremely parallelizable, since each grid element is independent of the rest and the entire grid can be updated simultaneously, in theory permitting a time complexity of $O(K)$. Although we precomputed the game-theoretic solution, our proposed computational method for the strategic planner can directly benefit from the ongoing advances in computer hardware for autonomous driving [219], so we expect that it will be feasible to compute the strategic value in an online setting.

Once the solution to the game has been computed, rather than attempting to *execute* any of the actions in this simplified dynamic representation, the autonomous vehicle can use the resulting value $V(s) := \max_a Q^0(s, a)$ as a guiding terminal reward term for the short-horizon trajectory planner.

8.2.2 Tactical planner: Open-loop trajectory optimization

In this section we demonstrate how to incorporate the strategic value into a low-level trajectory planner. We assume that the planner is performing a receding-horizon trajectory optimization scheme, as is commonly the case in state-of-the-art methods [220]. These methods tend to plan over relatively short time horizons (on the order of 1 s), continually generating updated "open-loop" plans from the current state—in most cases the optimization is local, and simplifying assumptions regarding the interaction are made in the interest of real-time computability.

While, arguably, strategic interactions can be expected to have a smaller effect over very short time-scales, the vehicle's planning should be geared towards efficiency and safety beyond the reach of a single planning window. The purpose of incorporating the computed

Algorithm 8.1: Feedback Stackelberg Dynamic Program

Data: $\hat{L}_A(\hat{s}, \hat{a}_A, \hat{a}_H),\ \hat{L}_H(\hat{s}, \hat{a}_A, \hat{a}_H)$
Result: $\hat{V}_A(\hat{s}, k)$, $\hat{V}_H(\hat{s}, k)$, $\hat{a}^*_A(\hat{s}, k)$, $\hat{a}^*_H(\hat{s}, k)$

Initialization
for $\hat{s} \in \hat{\mathcal{S}}$ **do**
A0 $\quad \hat{V}_A(\hat{s}, K+1) \leftarrow 0;$
H0 $\quad \hat{V}_H(\hat{s}, K+1) \leftarrow 0;$

Backward recursion
for $k \leftarrow K$ **to** 0 **do**
  for $\hat{s} \in \hat{\mathcal{S}}$ **do**
    for $\hat{a}_A \in \hat{\mathcal{A}}_A$ **do**
      for $\hat{a}_H \in \hat{\mathcal{A}}_H$ **do**
H1 $\quad\quad q_H(\hat{a}_H) \leftarrow \hat{L}_H(\hat{s}, \hat{a}_A, \hat{a}_H) + \hat{V}_H(\phi(\hat{s}, \hat{a}_A, \hat{a}_H), k+1);$
H2 $\quad P(\hat{a}_H \mid \hat{a}_A) \leftarrow \pi_H[q_H](\hat{a}_H);$
H3 $\quad q^*_H(\hat{a}_A) \leftarrow \sum_{\hat{a}_H} P(\hat{a}_H \mid \hat{a}_A) \times q_H(\hat{a}_H);$
A1 $\quad q_A(\hat{a}_A) \leftarrow \sum_{\hat{a}_H} P(\hat{a}_H \mid \hat{a}_A) \times \Big(\hat{L}_A(\hat{s}, \hat{a}_A, \hat{a}^*_H(\hat{a}_A)) + \hat{V}_A(\phi(\hat{s}, \hat{a}_A, a^*_H(\hat{a}_A)), k+1)\Big);$
A2 $\quad \hat{a}^*_A(\hat{s}, k) \leftarrow \arg\max_{\hat{a}_A} q_A(\hat{a}_A);$
A3 $\quad \hat{V}_A(\hat{s}, k) \leftarrow q_A(\hat{a}^*_A(\hat{s}, k));$
H4 $\quad \hat{a}^*_H(\hat{s}, k) \leftarrow a^*_H(\hat{a}^*_A(\hat{s}, k));$
H5 $\quad \hat{V}_H(\hat{s}, k) \leftarrow q^*_H(\hat{a}^*_A(\hat{s}, k));$

strategic value is to guide the trajectory planner towards states from which desirable long-term performance can be achieved.

We therefore formalize the tactical trajectory planning problem as an optimization with an analogous objective to (8.2) with a shorter horizon $M << N$ and instead introduce the strategic value as a terminal term representing an estimate of the optimal reward-to-go between $t = M$ and $t = N$:

$$R_A(x^{0:M}, u_A^{0:M}, u_H^{0:M}) = \sum_{t=0}^{M} L_A(x^t, u_A^t, u_H^t) + V_A\big(g(x^t)\big) \ . \tag{8.7}$$

The only modification with respect to a standard receding-horizon trajectory optimization problem is the addition of the strategic value term. Using the numerical grid computation

presented earlier, this can be implemented as an efficient look-up table, allowing fast access to values and gradients (numerically approximated directly from the grid).

The low-level optimization of (8.7) can thus be performed online by a trajectory optimization engine, based on some short-term predictive model of human decisions conditioned on the state and actions of the autonomous vehicle. In our results we implement trajectory optimization similar to [209] through a quasi-Newton scheme [221], in which the autonomous vehicle iteratively solves a nested optimization problem by estimating the human's best trajectory response to each candidate plan for the next M steps. We assume that the human has an analogous objective to the autonomous system, and can also estimate her strategic long-term value. We stress, however, that our framework is more general, and in essence agnostic to the concrete low-level trajectory optimizer used, and other options are possible (e.g. [19, 186]).

8.3 Simulation Results

We analyze the benefit of solving the dynamic game by comparing the proposed hierarchical approach to using a tactical planner only, as in the state of the art [186, 209]. We then compare against extended-horizon trajectory planning with an assumed open-loop information structure, showcasing the importance of reasoning with the fully coupled closed-loop feedback information of the dynamic game.

8.3.1 Implementation Details

Environment and Objective

We use a simulated two-lane highway environment with an autonomous car and a human-driven vehicle. Similar to [209], both vehicles' rewards encode safety and performance features through a number of additive terms. In particular, for the purposes of these case studies:

- Both players have a preference for driving along the center of a lane (squared-exponential reward terms).
- Both players prefer the left lane over the right lane (larger reward coefficient).
- Both players receive strong penalties for leaving the road (sigmoid cost).
- Both players receive strong penalties for colliding or coming to very close distance (squared-exponential cost).
- The autonomous car is given a target speed slightly faster than the human's (quadratic cost) and a preference for being ahead of her (sigmoid reward).

Tactical Level

The dynamics of each vehicle are given by a dynamic bicycle model with states $[\mathrm{x}_i, \mathrm{y}_i, v_i, \theta_i]$ (position, speed, and heading). The planner uses a discrete time step $\Delta t = 0.1$ s and $M = 5$ time steps. For the tactical trajectory planning, we compute the partial derivative $\frac{\partial R_i}{\partial \mathbf{u}_i}$ for each player and allow the optimization to proceed by iterated local best response between candidate autonomous vehicle plans and predicted human trajectories. If convergence is reached, the result is a local (open-loop) Nash equilibrium between the short-horizon trajectories [212, 214, 222].

Strategic Level

The full joint human-autonomous state space is 8-dimensional, making dynamic programming challenging. Our strategic level simplifies the state and dynamics using an approximate, lower-order representation. We consider a larger time step of $\Delta k = 0.5$ s and a horizon $K = 10$ corresponding to 5 s. We consider one of two high-level models, depending on the setup.

Two-vehicle setup. If the environment is a straight empty highway, it is enough to consider the longitudinal position of the two vehicles relative to each other: $\mathrm{x}_{\mathrm{rel}} = \mathrm{x}_A - \mathrm{x}_H$. We assume the human-driven vehicle's average velocity is close to the nominal highway speed 30 m/s, and the vehicles' headings are approximately aligned with the road at all times. Finally, given the large longitudinal velocity compared to any expected lateral velocity, we assume that vehicles can achieve any desired lateral velocity up to ± 2.5 m/s within one time step (consistent with a typical 1.5 s lane change). The approximate dynamics are then

$$[\dot{x}_{\mathrm{rel}},\ \dot{y}_A,\ \dot{y}_H,\ \dot{v}_{\mathrm{rel}}] = [v_{\mathrm{rel}},\ w_A,\ w_H,\ a_A - a_H - \tilde{\alpha} v_{\mathrm{rel}}]\ , \tag{8.8}$$

with the control inputs being the vehicles' lateral velocities w_A, w_H and accelerations a_A, a_H, and where $\tilde{\alpha}$ is the linearized friction parameter. This allows us to implement Algorithm 8.1 on a $75 \times 12 \times 12 \times 21$ grid and compute the feedback Stackelberg solution of the strategic game.

Additional-vehicle setup. If there are additional vehicles or obstacles present in the environment, it becomes necessary to explicitly consider absolute positions and velocities of the two players' vehicles (or at least relative to these other objects). In this scenario, we consider a truck driving in the right lane at a constant speed v_T, and assume that the human remains in her lane. Letting x_{AT}, x_{HT} denote the longitudinal position of each vehicle relative to the truck and α be the friction coefficient, the high-level dynamics are

$$\begin{aligned}[\dot{x}_{AT}, \dot{\mathrm{x}}_{HT}, \dot{\mathrm{y}}_A, \dot{v}_A, \dot{v}_H] = \\ [v_A - v_T,\ v_H - v_T,\ w_A,\ a_A - \alpha v_A^2,\ a_H - \alpha v_H^2]\,.\end{aligned} \tag{8.9}$$

We implement Algorithm 8.1 on a $35 \times 35 \times 6 \times 8 \times 8$ grid.

Human simulation.

For consistency across our case studies, we simulated the human driver's behavior using the same model throughout. We assume that the human driver makes accurate predictions of the autonomous vehicle's imminent trajectory for 0.5 s. While we found that for the maneuvers considered a low-level trajectory optimizer produced sufficiently realistic driving behavior, the results presented here should be taken as a preliminary proof of concept, and proper validation with human drivers would be a necessary next step to more reliably determine the advantages of the hierarchical game-theoretic planning scheme.

8.3.2 Interaction Case Studies

We compare the tactical-only trajectory planner (baseline) against the proposed hierarchical tactical-strategic planning scheme for 3 different driving scenarios.

Merge maneuvers

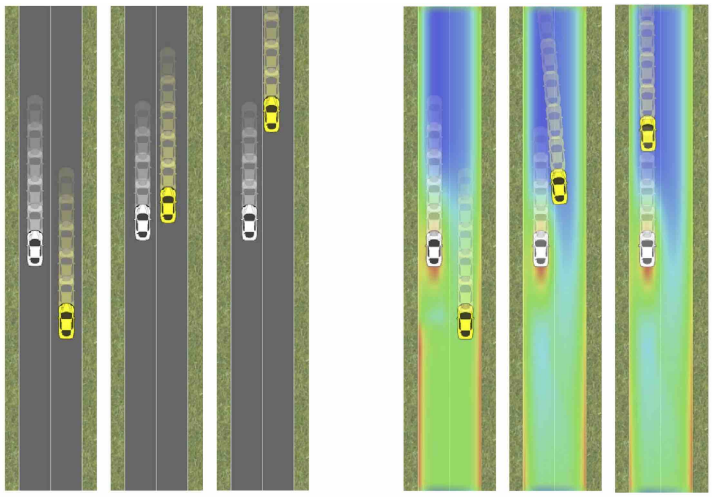
We begin with a simple merge maneuver where the autonomous vehicle's objective rewards it for driving on the left lane and its target speed is faster than the human's.

starts *ahead* of the human in the adjacent lane. The tactical planner leads the autonomous car to successfully merge in front of the human. The hierarchical planner also succeeds, with the strategic value guiding the vehicle to merge more swiftly, improving performance.

Next, we consider the case where the autonomous car starts *behind* the human, as depicted in Figure 8.2. The tactical autonomous car accelerates ahead of the human but does not merge into her lane (likely due to a local optimum in the trajectory optimization). The hierarchical autonomous car overtakes and merges in front of the human.

Overtaking

We now study a complete overtaking maneuver in which the autonomous car starts behind the human in the *same* lane. The tactical autonomous car does not successfully complete the maneuver: it first accelerates but then brakes to remain behind the human, oblivious to the higher long-term performance achievable through overtaking. The hierarchical planner produces a policy that, depending on the human's behavior, can evolve into two alternative strategies, shown in Figure 8.1. First, the autonomous vehicle approaches the human from behind, expecting her to have an incentive (based on her strategic value) to change lanes and let it pass. If this initial strategy is successful and the human changes lanes, the autonomous vehicle overtakes without leaving the left lane. Conversely, if the human does not begin a lane change, the strategic value guides the autonomous vehicle to merge into the right lane, accelerate to overtake the human, reaching a maximum speed of 37.83 m/s (2.83 m/s above its target speed), and merge back into the original lane.



(a) Low-level planner (b) Hierarchical planner

Figure 8.2: Tactical versus hierarchical trajectory planning in a highway merging maneuver. The tactical (low-level) trajectory planner gets ahead of the human-driven vehicle but does not merge into the left lane. The game-theoretic hierarchical planner merges in front of the human, guided by the strategic value.

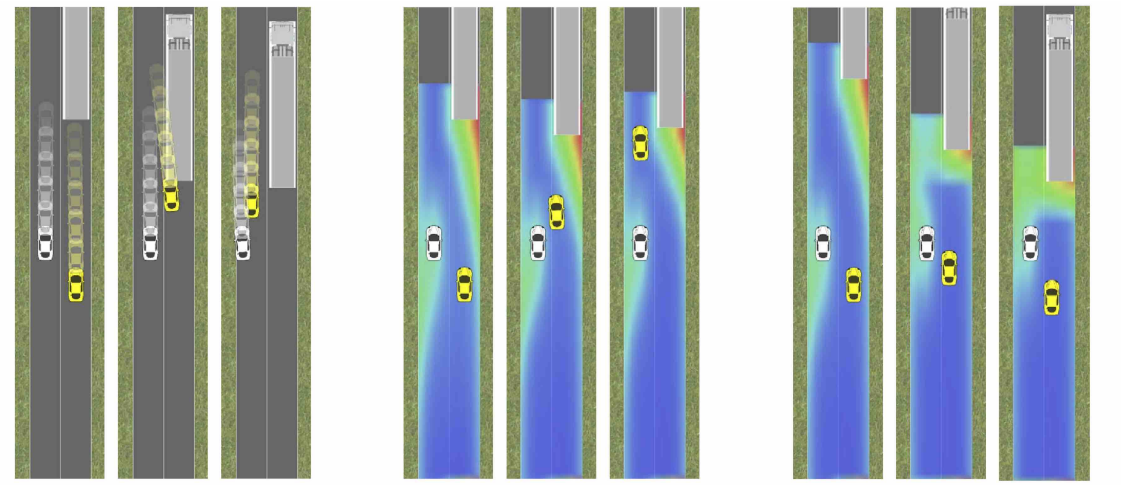
Truck Cut-In

Finally we consider a scenario in which the two vehicles are approaching a truck, assumed to drive at a lower constant speed of 26.82 m/s. As shown in Figure 8.3, the tactical-only planner may attempt merges with little safety margin. The hierarchical game-theoretic analysis allows us to reason through the leverages players may have on each other. If the autonomous vehicle has a sufficient initial speed, the human is incentivized to slow down to allow it to merge safely in front of her before reaching the truck. Otherwise, she will instead accelerate, incentivizing the autonomous car to slow down, abort the overtaking maneuver, and merge behind her instead to pass the truck safely.

Note that we are not proposing that autonomous vehicles should in fact carry out this type of overtaking maneuver. The remarkable result here is in the planner's ability to reason about the different possible strategies given the scenario and objectives. Also note that in this and the other example scenarios, the roles of the human and the autonomous vehicle can easily be interchanged, allowing the autonomous vehicle to e.g. discourage others' potentially unsafe maneuvers.

8.3.3 Additional Analysis

We now seek to shed light on *why* hierarchical planning obtains better performance than tactical alone. Is the strategic value merely lengthening the effective horizon, avoiding local or myopic optima, or is information structure important?



(a) Low-level planner (b) Hierarchical planner: human lets autonomous vehicle cut in (c) Hierarchical planner: human cuts off autonomous vehicle

Figure 8.3: Tactical versus hierarchical planning in a cut-in scenario while overtaking a truck. (a) The tactical-only planner executes an unsafe last-second merge. (b) With enough speed difference, the hierarchical planner first accelerates to incentivize the human to slow down and then safely merges in front. (c) If there is little margin, the human has an incentive to accelerate preventing the maneuver.

Hierarchical vs. long-horizon tactical planning

The hierarchical planning method provides the autonomous car with more information about the future via the strategic value of the long-term game, which guides the optimization to escape local optima. If those were the only benefits, extending the horizon of the tactical planner and re-initializing in different basins of attraction ought to perform similarly. We thus extend the horizon to 2 s (20 time steps) and perform multiple independent optimizations at each planning cycle, initialized from diverse trajectories for each car: full-left steer, full-right steer, and straight steer (with acceleration input to maintain speed). This stronger tactical planner is unable to optimize in real time, unlike our other demonstrations, but is a good tool for analysis. Extension beyond 2 s was not tractable.

We tested this planner in the overtaking scenario alongside a human-driven car that is aware of the autonomous car's plan, which is this planner's assumed information structure. The planner still fails to complete the maneuver regardless of the initialization scheme and whether the influence term in [209] is used, resulting in the autonomous car remaining behind the human, as shown in Figure 8.4. Moreover, we tested this planner against a human driver who maintains a constant slow speed of 24 m/s. In this case, the autonomous car brakes abruptly to avoid a collision and remains behind the human, at each time step expecting her to maximally accelerate for the next 1 s. Despite the longer horizon and more global

optimization, this new tactical planner still assumes the wrong information structure, i.e. that the human knows the autonomous car's trajectory multiple seconds into the future. This causes poor performance when the human does not in fact adapt to the autonomous vehicle's plan ahead of time.

Information structure at the tactical level

When optimizing the autonomous car's trajectory at the tactical level, we used iterated local best response seeking a local open-loop Nash equilibrium between the vehicles' short-horizon trajectories. Conversely, the implicit differentiation proposed in [209], by which the autonomous planner estimates the influence of each local trajectory change on human's best response, is consistent with the local open-loop Stackelberg equilibrium concept, with the human as the *follower*. We observed that this latter approach resulted in more aggressive behavior in some situations, even when augmenting this tactical planner with the long-term strategic value. For example, in the hard merge scenario shown in Figure 8.4, the hierarchical car attempted to merge into the left lane before fully overtaking the human, placing the burden on her to avoid an imminent collision. On the other hand, the traditional "pipeline" approach, in which the human's trajectory is predicted and fed to the planner as a moving obstacle, failed to overtake when used by itself, but succeeded in changing lanes and overtaking (comparably to the iterated best response scheme) when given the strategic value term.

The results suggest that, even in short horizons, assuming that the human can accurately anticipate and adapt to the autonomous vehicle's planned trajectory may lead to unsafe situations when the actual human driver fails to preemptively make way as expected. Running iterated local best response between trajectories or even assuming no short-term human adaptation at the tactical level seem to perform better as tactical schemes within our proposed hierarchical framework.

Confidence in Strategic Human Model

Finally, we discuss the effects of varying the autonomous planner's confidence in its high-level model of the human. Following the approach in Chapter 7, it is possible for the autonomous vehicle to reason online about the reliability of its model of the human based on its ability to accurately describe her ongoing behavior.

Modeling the human as a Boltzmann noisily rational agent, we can readily combine the probabilistic policy (quantal response) model (8.5) with the Bayesian confidence update (7.9) proposed in Chapter 7. Given a prior b^0 on the inverse temperature coefficient β (conceivably obtained from the increasingly available driving datasets [223, 224]), the system can maintain an updated belief on a particular human agent by recurrently applying the update:

$$b^{t+1}(\beta) = \frac{P(a_H \mid s, a_A; \beta, \theta) b^t(\beta)}{\sum_{\hat{\beta}} P(a_H \mid s, a_A; \hat{\beta}, \theta) b^t(\hat{\beta})} \ . \tag{8.10}$$

This enables the autonomous vehicle to reason in real time about how accurately its game-theoretic model of the human predicts her interactions, and adapt its game-theoretic planning accordingly.

As in Chapter 7, it is in principle possible to simultaneously update a belief on model confidence and other model parameters, which could encode the human's internal state; some work in this direction has shown promising results (cf. [210]). Here, we focus on investigating the effect of varying the vehicle's confidence in a given human model, which should be seen as complementary to any possible parameter updates.

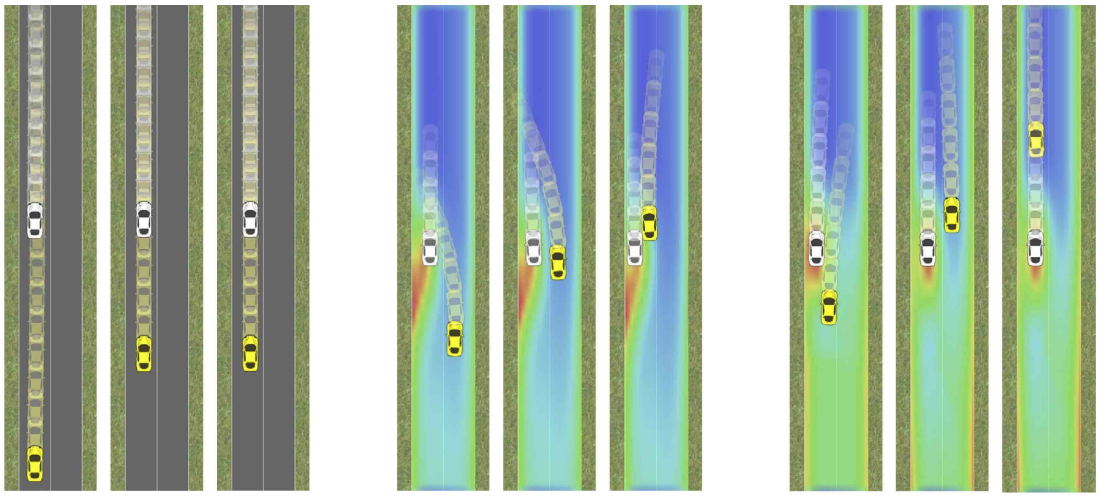
We compute different strategic values corresponding to varying levels of confidence in the human model. In the overtaking scenario, we observe that inverse temperature parameters below a certain threshold lead to the autonomous vehicle choosing to remain behind the human-driven car instead of attempting to overtake. A lower level of confidence in the human model discourages the autonomous car from overtaking because the human driver is seen as less likely to respond to the autonomous vehicle's actions consistently with the modeled strategic incentives and more likely to act in a poorly understood fashion that may result in a collision.

8.4 Chapter Summary

We have introduced a hierarchical trajectory planning formulation for an autonomous vehicle interacting with a human-driven vehicle on the road. To tractably reason about the mutual influence between the human and the autonomous system, our framework uses a lower-order approximate dynamical model solve a nonzero sum game with closed-loop feedback information. The value of this game is then used to inform the planning and predictions of the autonomous vehicle's low-level trajectory planner.

Even with a simplified dynamical model, solving the dynamic game will generally be computationally intensive. We note, however, that our high-level computation presents two key favorable characteristics for online usability. First, it is "massively parallel" in the sense that all states on the discretized grid may be updated simultaneously. The need for reliable real-time perception in autonomous driving has spurred the development of high-performance parallel computing hardware, which will directly benefit our method. Second, once computed, the strategic value can be readily stored as a look-up table, enabling fast access by the low-level trajectory planner. Of course, strategic values would then need to be pre-computed for a number of scenarios that autonomous vehicles might encounter.

Overall, hierarchical game-theoretic reasoning schemes like the one investigated here may work in conjunction with and significantly enhance existing autonomous driving planners, and may play a significant role in allowing autonomous vehicles to safely and efficiently interact with human road users.



(a) Long-horizon tactical-only planner

(b) Hierarchical planner with low-level influence estimate

(c) Hierarchical planner with predict-then-plan low-level scheme

Figure 8.4: Study of alternative information structures. (a) In the overtaking scenario, the long-horizon tactical-only car accelerates, expecting the human to match its higher speed to avoid a collision. After the human speeds up, the autonomous car remains behind her. (b) Under the influence estimate [209] in the low-level trajectory gradient, the hierarchical car drives more aggressively in the merging scenario. (c) When augmented with the strategic value, the "pipeline" (predict-then-plan) low-level scheme is able to overtake.

Part III

Safe Steps Forward

Chapter 9

Safety Analysis through Reinforcement Learning

The greatest teacher failure is.

Master Yoda
The Last Jedi, 2017

This chapter is based on the paper "Bridging Hamilton-Jacobi Safety Analysis and Reinforcement Learning" [225], written in collaboration with Neil Lugovoy, Vicenç Rubies Royo, Shromona Ghosh, and Claire Tomlin.

In this final technical chapter, we offer a look forward by revisiting the idea of synergy between safety and learning, which we first introduced in Chapter 6, in the context of the increasingly powerful data-driven methods developed within the artificial intelligence community. By establishing a theoretical and algorithmic connection between control-theoretic safety analysis and modern reinforcement learning formulations, we explore a potentially fruitful avenue for future research, which has the potential to enable scalable safety analysis for complex, high-dimensional systems.

In recent years, reinforcement learning techniques have proven their usefulness in computing data-driven approximate solutions to optimal control problems seeking the maximization of a discounted additive payoff in complex and high-dimensional systems [148, 149, 226, 227]. Unfortunately, objective functions representing a sum of rewards over time are not well suited to capture the safety objective, since, as we have seen extensively throughout this dissertation, safety is not determined by how much a system fails on average, but by whether it fails *at all*. Partly for this reason, reinforcement learning techniques have not seen widespread use for safety analysis.

Another consequence of this disconnect between formulations is that controllers computed through reinforcement learning are typically not inherently safety-preserving, a limitation that, as we discussed in Chapter 6, has hindered their applicability to physical autonomous

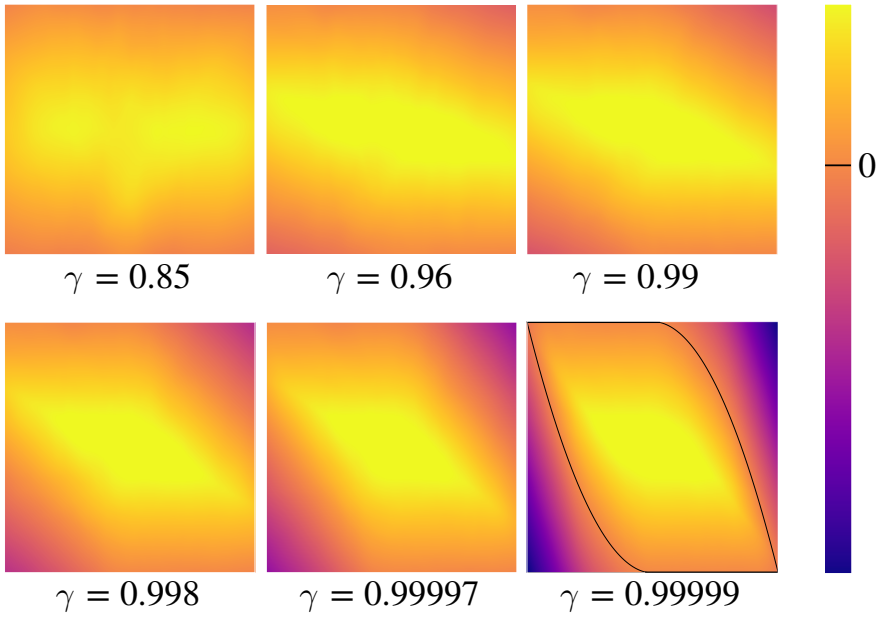


Figure 9.1: Multiple snapshots of the neural network output of the Safety Q-learning algorithm for a double-integrator system. As we anneal the discount factor $\gamma \to 1$ during Q-learning, the learned discounted safety value function asymptotically approaches the undiscounted value, allowing us to recover the safe set and optimal safety policy with very high accuracy.

systems. Recent years have seen a growing interest in "safe learning" schemes. While some recent approaches have proposed approximating safety by stability [140, 161] or bounded rates of constraint violation [137, 138], Hamilton-Jacobi analysis stands out as a powerful approach to rigorously ensure robust constraint satisfaction during learning. As we have seen throughout Parts I and II of this thesis, however, this family of methods inherits the scalability challenges of numerical dynamic programming computations beyond low-dimensional systems.

The work in this chapter seeks to unlock a new family of tools for safety analysis by rendering a wide range of state-of-the-art methods in the reinforcement learning literature readily usable for safety analysis in high-dimensional systems. By introducing a time discount into the dynamic programming equation of Hamilton-Jacobi safety analysis, we can obtain a contraction Bellman operator that lends itself to the use of temporal difference learning techniques. We prove the key properties of this discounted *Safety Bellman Equation* and show that it can be readily used with value-learning approaches from the reinforcement learning literature: in particular, we prove that the resulting *Safety Q-learning* scheme converges to the safety state-action value function in finite Markov decision processes.

The Safety Q-learning scheme allows us to recover the globally optimal solution to the corresponding Hamilton-Jacobi analysis (to resolution completeness [228, 229]) in low-dimensional problems where dense computation is viable: we validate the results using tab-

ular Q-learning against a double integrator system, achieving high accuracy relative to the analytic solution. Crucially, we further observe comparable performance when replacing the state-space grid with a neural network function approximator, which is key in order to scale to higher dimensions. Finally, annealing the discount factor during learning allows asymptotic recovery of the solution to the *undiscounted* safety problem (Figure 9.1).

We evaluate deep Safety Q-learning on a variety of simulated robotics tasks, and observe consistently accurate results against numerical dynamic programming solutions. In high-dimensional systems beyond the reach of traditional numerical methods, predicted safety accurately matches the empirical performance of the learned safety controller.

We finally implement policy optimization through an adaptation of the basic REINFORCE algorithm [230] to the discounted safety formulation, and explore the potential of using sate-of-the-art methods by similarly adapting the soft actor-critic (SAC) scheme [231]. The promising results on an 18-dimensional problem suggest the usability of this family of reinforcement learning methods for learning policies with the ability to preserve safety in high-dimensional systems.

It is important to clarify that, while the formulation in this chapter yields a promising new tool for safety analysis, it is not—by itself—a safe learning framework, since it requires *experiencing failures* in order to learn about safety. Therefore, the approach introduced here should primarily be thought of as a computational tool, alternative to other methods like the numerical Hamilton-Jacobi calculations [51, 52] used throughout this thesis. That is, the safety-learning method presented in this chapter will typically be used in conjunction with a model (simulation) of the system dynamics, as is commonly done with most "model-free" reinforcement learning methods. In principle, it would be possible to instead implement the scheme in a *truly* model-free fashion, learning to ensure constraint satisfaction directly on the physical system, as long as the training is limited to conditions that are not truly safety-critical, such as a vehicle test track with only virtual obstacles.

Once the safety analysis has been computed (learned), whether in simulation or under non-critical training conditions, the resulting control policy can be applied to the physical system in similar conditions to other safety controllers. This means that we can certainly utilize this method as *part* of a safe learning framework: the reinforcement learning scheme in this chapter can be used in lieu of the numerical safety computation in Chapter 6, complemented with some additional decision logic that can translate the learned best-effort safety policy into high-confidence recursive feasibility guarantees in the vein of the safe exploration scheme in Chapter 5, Section 5.2. Thus, we predict that further investigation of the questions opened by this chapter will be fruitful in obtaining scalable least-restrictive supervisory control schemes for learning-based controllers to improve their performance while maintaining safety.

Finally, we note that while the analysis throughout the chapter is presented for deterministic dynamics, robust and stochastic extensions are possible (and have been explored to some extent in [232]). Such extensions will be important for implementation of the formulation on physical systems, which is of course its ultimate intended application.

9.1 The Undiscounted Safety Problem

As in previous chapters, we consider a dynamical system with state $x \in \mathbb{R}^n$, control input $u \in \mathcal{U} \subset \mathbb{R}^{n_u}$, and continuous-time dynamics

$$\dot{x} = f(x, u) \ . \tag{9.1}$$

with the usual technical assumptions ensuring existence of Carathéodory solutions.

We consider a closed constraint set $\mathcal{K} \subset \mathbb{R}^n$, and define a bounded function $g : \mathbb{R}^n \to \mathbb{R}$ that is satisfies $g(x) \geq 0 \iff x \in \mathcal{K}$. Then, the value function

$$V(x) := \sup_{\mathbf{u}(\cdot)} \inf_{t \geq 0} g\big(\mathbf{x}_x^{\mathbf{u}}\big) \tag{9.2}$$

captures the minimum safety margin g achieved over time by a trajectory starting at each state $x \in \mathbb{R}^n$ if the best possible control input is applied at every instant. Recall that this can intuitively be thought of as the *closest* the system will get to violating the constraints, as measured by the "signed distance" g.

As we know, this minimum-payoff optimal control problem can be approached via dynamic programming. Considering a finite time horizon $t \in [0, T]$, it is possible to compute the optimal safety value function as the solution to a time-dependent terminal-value Hamilton-Jacobi-Bellman variational inequality of the same form as (6.6):

$$\begin{aligned} 0 &= \min \left\{ g(x) - V(x,t), \frac{\partial V}{\partial t} + \max_{u \in \mathcal{U}} \nabla_x V^\top f(x,u) \right\} \ , \\ V(x,T) &= g(x) \ , \quad \forall x \in \mathbb{R}^n \ . \end{aligned} \tag{9.3}$$

The discrete-time counterpart, following Algorithm 3.1 in Chapter 3, will be the starting point for our discounted formulation. Given a discrete time step δ, the value at time t can be recursively computed from the value at time $t + \delta$ as:

$$V(x,t) = \min \left\{ g(x), \max_{u \in \mathcal{U}} V\big(x + f(x,u)\delta, t + \delta\big) \right\} \ . \tag{9.4}$$

In the infinite-horizon case, the value function no longer changes in finite time, and so $V(x)$ must satisfy the fixed-point Bellman equation:

$$V(x) = \min \left\{ g(x), \max_{u \in \mathcal{U}} V\big(x + f(x,u)\delta\big) \right\} \ . \tag{9.5}$$

An important observation about (9.5) is that, unlike the discounted Bellman equation (2.50) commonly used in reinforcement learning, it does *not* induce a contraction mapping on V and therefore it is not generally possible to converge to the correct value function by application of value iteration or temporal difference learning.

In the next section we introduce a modification of (9.5) that yields a contraction mapping for our problem of interest, and extend the convergence results of temporal difference learning to safety control problems.

9.2 The Discounted Safety Bellman Equation

The central contribution of this chapter is a modified form of the dynamic programming safety backup (9.5) which induces a contraction mapping in the space of value functions and is therefore amenable to reinforcement learning methods based on temporal difference learning [68, 148, 233].

The key observation stems from an intuitive interpretation of time-discounting in the problem of cumulative rewards: at every instant, there is a small probability $1-\gamma$ of transitioning to an absorbing state from which no more rewards will be accrued. Thus, in the corresponding trajectory outcome

$$\mathcal{V}(\mathbf{x}_x^{\mathbf{u}}) = \sum_{k=0}^{K} \gamma^k L\big(x_k, u_k\big) \ , \tag{9.6}$$

the discount factor $\gamma \in [0,1)$ can be seen as the probability of the episode continuing, with $1-\gamma$ conversely representing the probability of transitioning to a terminal state.

An analogous interpretation in the problem of minimum payoff over time can be achieved by modifying (9.5) to account for such a transition. Here if, with probability $1-\gamma$, an episode were to end after the current time step, the minimum future $g(\cdot)$ would be equal to the current $g(x)$. This induces the discrete-time discounted dynamic programming equation

$$V(x) = (1-\gamma)g(x) + \gamma \min\Big\{g(x), \max_{u\in\mathcal{U}} V\big(x + f(x,u)\Delta t\big)\Big\}. \tag{9.7}$$

Letting g_i be the value of g achieved by a discrete-time state trajectory $\mathbf{x}_x^{\mathbf{u}}$ at the i-th time step, the explicit form of the objective maximized in (9.7) is a "time-discounted" minimum:

$$\begin{aligned}\mathcal{V}(\mathbf{x}_x^{\mathbf{u}}) = (1-\gamma)g_0 + \gamma\Big[\min\big\{g_0, (1-\gamma)g_1 + \\ \gamma(\min\{g_1, (1-\gamma)g_2 + \gamma\ldots)\}\big\}\Big] \ .\end{aligned} \tag{9.8}$$

We prove two key properties of the proposed equation.

Theorem 9.1. *(Contraction mapping) The discounted Safety Bellman Equation* (9.7) *induces a contraction mapping under the supremum norm. That is, let $V, \tilde{V} : \mathbb{R}^n \to \mathbb{R}$, then there exists a constant $\kappa \in [0,1)$ such that $\|B[V] - B[\tilde{V}]\|_\infty \leq \kappa \|V - \tilde{V}\|_\infty$.*

Proof. It will suffice to show that for *all* states $x \in \mathbb{R}^n$, $|B[V](x) - B[\tilde{V}](x)| < \kappa\|V - \tilde{V}\|_\infty$. We have:

$$\begin{aligned}&|B[V](x) - B[\tilde{V}](x)| \\ &\quad = \gamma|\min\{g(x), \max_{u\in\mathcal{U}} V\big(x + f(x,u)\Delta t\big)\} \\ &\qquad - \min\{g(x), \max_{\tilde{u}\in\mathcal{U}} \tilde{V}\big(x + f(x,\tilde{u})\Delta t\big)\}| \\ &\quad \leq \gamma|\max_{u\in\mathcal{U}} V\big(x + f(x,u)\Delta t\big) - \max_{\tilde{u}\in\mathcal{U}} \tilde{V}\big(x + f(x,\tilde{u})\Delta t\big)| \ .\end{aligned}$$

Now, without loss of generality suppose the first maximum is the larger one, and let $u^* \in \mathcal{U}$ achieve it. We continue:

$$\begin{aligned}
&|B[V](x) - B[\tilde{V}](x)| \\
&\quad \leq \gamma |V\big(x + f(x,u^*)\Delta t\big) - \tilde{V}\big(x + f(x,u^*)\Delta t\big)| \\
&\quad \leq \gamma \max_{u \in \mathcal{U}} |V\big(x + f(x,u)\Delta t\big) - \tilde{V}\big(x + f(x,u)\Delta t\big)| \\
&\quad \leq \gamma \sup_{\tilde{x}} |V(\tilde{x}) - \tilde{V}(\tilde{x})| = \gamma \|V - \tilde{V}\|_\infty \ .
\end{aligned}$$

Thus the sought contraction constant is in fact $\gamma \in [0, 1)$. □

Proposition 9.1. *(Value approximation) In the limit of no discounting, the fixed-point solution to the Safety Bellman Equation* (9.7) *converges to the undiscounted safety value function.*

Proof. Taking the limit of the optimization of (9.8) as γ goes to 1 we recover:

$$\lim_{\gamma \to 1} V(x) = \max_{\mathbf{u}^{0:T}} \min \big\{ g_0, g_1, g_2, \dots \big\} \ ,$$

which solves (9.5) and is the discrete-time approximation to (9.2). □

The above two theoretical results enable the use of reinforcement learning techniques for safety analysis. We end this section with an important consequence of Theorem 9.1.

Theorem 9.2. *(Convergence of Safety Q-learning) Let $\mathbf{X} \subseteq \mathbb{R}^n$ and $\mathbf{U} \subseteq \mathcal{U}$ be finite discretizations of the state and action spaces, and let $\mathbf{f} : \mathbf{X} \times \mathbf{U} \to \mathbf{X}$ be a discrete transition function approximating the system dynamics. The Q-learning scheme applied to the discounted safety problem and executed on the above discretization converges, with probability* 1*, to the optimal state-action safety value function*

$$Q(x_\mathbf{i}, u_\mathbf{j}) := (1-\gamma) g(x_\mathbf{i}) + \gamma \min \Big\{ g(x_\mathbf{i}), \max_{u_\mathbf{j}' \in \mathbf{U}} Q\big(\mathbf{f}(x_\mathbf{i}, u_\mathbf{j}), u_\mathbf{j}'\big) \Big\} \ ,$$

in the limit of infinite exploration time and given partly-random episode initialization and learning policy with full support over $\mathbf{X}$ and $\mathbf{U}$ respectively. Concretely, learning is carried out by the update rule:

$$\begin{aligned}
Q_{k+1}(x_\mathbf{i}, u_\mathbf{j}) \leftarrow Q_k(x_\mathbf{i}, u_\mathbf{j}) + \alpha_k \Big[&(1-\gamma) g(x_\mathbf{i}) + \\
&\gamma \min \Big\{ g(x_\mathbf{i}), \max_{u_\mathbf{j}' \in \mathbf{U}} Q\big(\mathbf{f}(x_\mathbf{i}, u_\mathbf{j}), u_\mathbf{j}'\big) \Big\} - Q_k(x_\mathbf{i}, u_\mathbf{j}) \Big] ,
\end{aligned}$$

for learning rate $\alpha_k(x_\mathbf{i}, u_\mathbf{j})$ satisfying

$$\sum_k \alpha_k(x_\mathbf{i}, u_\mathbf{j}) = \infty \quad \sum_k \alpha_k^2(x_\mathbf{i}, u_\mathbf{j}) < \infty \ ,$$

for all $x_\mathbf{i} \in \mathbf{X}, u_\mathbf{j} \in \mathbf{U}$.

Proof. The proof follows from the general proof of Q-learning convergence for finite-state, finite-action Markov decision processes presented in [234]. The transition dynamics $\mathbf{f}$, initialization and policy randomization, and learning rate α_k satisfy Assumptions 1, 2, and 3 in [234] in the standard way. The only critical difference in the proof is the contraction mapping, which we obtain under the supremum norm by Theorem 9.1: with this, Assumption 5 in [234] is met, granting convergence of Q-learning by Theorem 3 in [234]. □

We stress that, beyond Q-learning, the contraction-mapping property of the discounted safety backup opens the door to straightforward application of a wide variety of reinforcement learning schemes to safety analysis. We dedicate the following section to a first demonstration in which we explore the application of canonical reinforcement learning algorithms in the two main families: value learning and policy optimization.

9.3 Results

We present the results of implementing the proposed discounted Safety Bellman Equation in multiple reinforcement learning schemes: tabular Q-learning [68], deep Q-learning (DQN) [148], REINFORCE [230], and soft actor-critic (SAC) [231], and four different dynamical systems. We first validate the computed safety value function and safe set against analytically and numerically obtained ground-truth references in traditionally tractable systems. We consider two dynamical systems commonly used as benchmarks in control theory, namely a 2-D double-integrator system and a 4-D cart-pole system. We then demonstrate the scalability and usefulness of the formulation in higher-dimensional nonlinear systems, for which exact safety analysis is generally considered intractable. We use simulation environments common in reinforcement learning [235], namely a 6-D lunar lander system and an 18-D "half-cheetah" system.

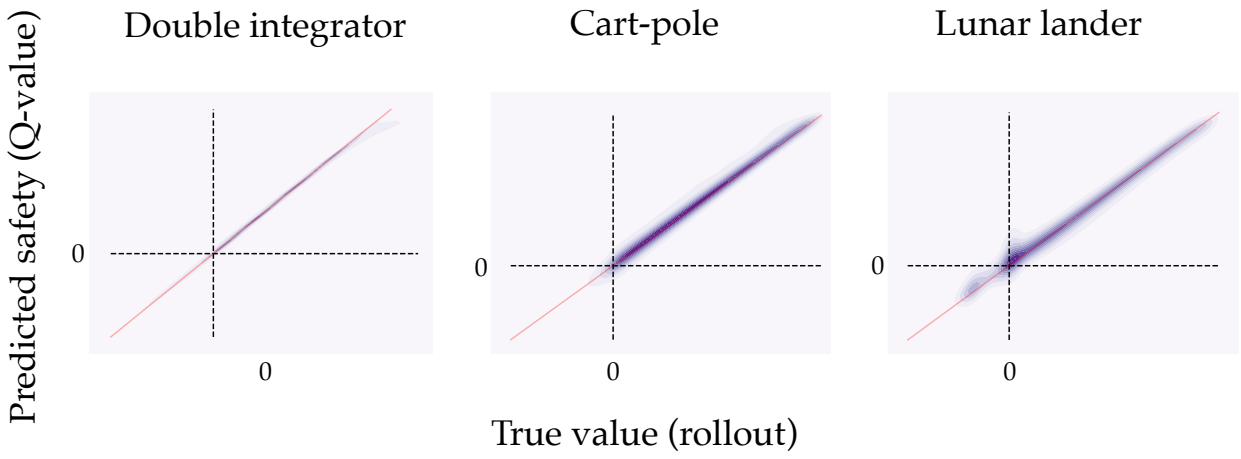


Figure 9.2: Predicted vs. achieved minimum safety margin to violations for 10^6 simulated rollouts with 100 trained Safety Q networks. Red line indicates identity.

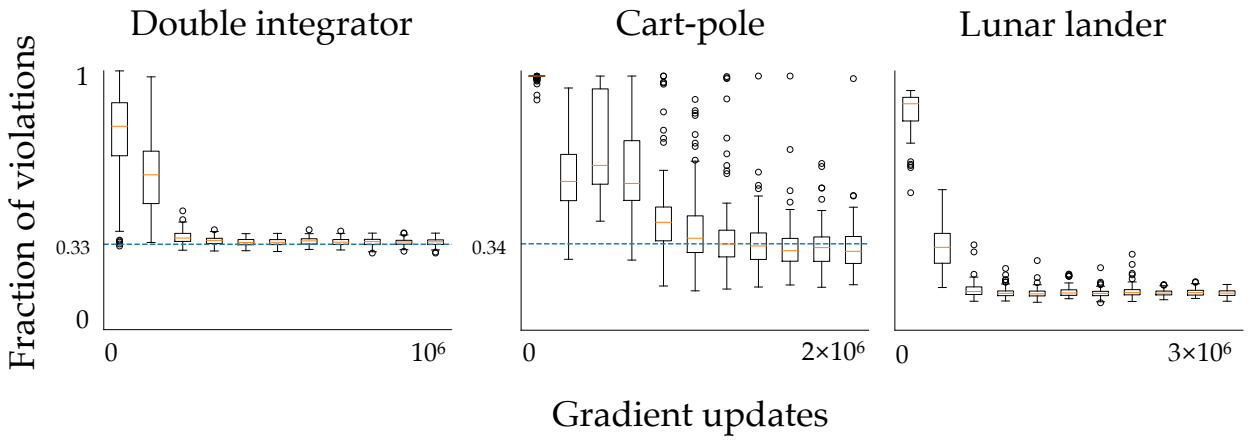


Figure 9.3: Fraction of initial conditions resulting in safety violations as Safety Q-Learning training proceeds. Each data point is a sample average from 1000 episodes; statistics are taken over 100 independent training runs. As learning progresses, the fraction of violations reliably decreases, approaching the ground-truth fraction of unsafe states (from which violation is inevitable) for the double integrator and cart-pole. Lunar lander ground truth is unknown.

9.3.1 Validation: comparison to ground truth

Analytic validation: double integrator

The double integrator is a classic reachability example where the control policy seeks to keep the system in the set $\{[\mathrm{x}, v] \in \mathbb{R}^2 : \mathrm{x} \in [\mathrm{x}_{\min}, \mathrm{x}_{\max}]\}$ with the dynamics characterized by:

$$\dot{\mathrm{x}} = v \ , \quad \dot{v} = u \ , \tag{9.9}$$

with $|u| \leq u_{\max}$, where x can be seen as position, v as velocity, and u as an acceleration input. Analytically, the safe set is characterized by the interior of the boundary defined by the parabolic segments

$$\begin{cases} \mathrm{x}_{\mathrm{low}} + \frac{v^2}{2u_{\max}} & v \leq 0 \\ \mathrm{x}_{\mathrm{high}} - \frac{v^2}{2u_{\max}} & v \geq 0 \end{cases} \tag{9.10}$$

and the boundaries $\mathrm{x} = \mathrm{x}_{\mathrm{low}}$, $\mathrm{x} = \mathrm{x}_{\mathrm{high}}$. Although simple, this example proves a useful context for visualizing the effect of γ, since the entire value function can be represented in two dimensions.

It can be seen in Figure 9.1 how as γ is annealed the time horizon of safety is effectively extended: for lower values the value function resembles $g(\cdot)$, and for higher values it approaches the undiscounted value function. Final accuracy and in-training performance are shown in Figure 9.2 and Figure 9.3.

Using tabular Q-learning with $g(\cdot)$ as the signed Euclidean distance to the boundary of the constraint set and annealing γ to 1 similar to [236], we observe convergence to the safe set up to the resolution of the grid. Independently training 100 deep Q-networks [148]

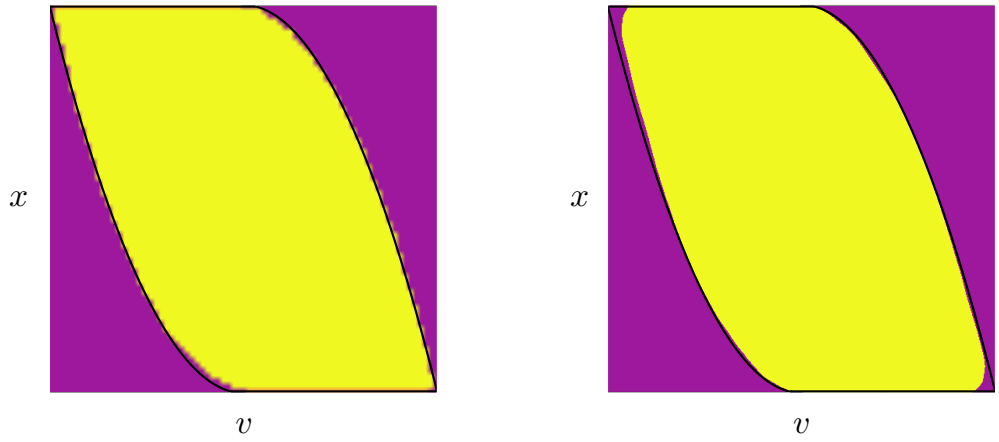


Figure 9.4: Safe sets learned by tabular (left) and deep Safety Q-Learning (right) with the discounted Safety Bellman Equation compared to the analytic set (black).

with fully-connected layers using the discounted Safety Bellman Equation we find near-convergence to the safe set with an average 2.26×10^{-5} (minimum 0, maximum 1.27×10^{-4}) fraction of points incorrectly characterized as safe and an average 1.76×10^{-4} (minimum 3.26×10^{-5}, maximum 3.31×10^{-4}) of points falsely characterized as unsafe. Classification is visualized in Figure 9.4.

Numerical validation: cart-pole

The cart-pole system (inverted pendulum) is a classic control problem and one ripe for safety analysis. A cart moving on a one-dimensional track is attached by a pivot to a pole. The control policy seeks to keep the pole from falling and to keep the cart from the edge of the track by applying accelerations to the cart. For this system, the ground truth safe set must be computed numerically on a grid using dynamic programming [49]. Over 100 random seeds we find that an average 5.16×10^{-5} (minimum 4.90×10^{-6}, maximum 2.56×10^{-4}) fraction of points are misclassified as safe and an average fraction 5.80×10^{-4} (minimum 4.24×10^{-4}, maximum 8.4×10^{-4}) fraction of points are misclassified as unsafe, relative to the numerically approximated ground truth. In reality, the precision of the numerical ground truth is limited by the grid resolution; thus, if we consider any points less than one full grid cell away from a safe grid point to be safe, we find that only an average 1.47×10^{-6} (minimum 0, maximum 4.54×10^{-5}) fraction of points are misclassified as safe by the proposed method (Figures 9.2 and 9.3).

9.3.2 Scalability: safety for high dimensional systems

The two examples we have shown thus far help us validate the proposed approach against well-established safety analysis tools. However, a motivating factor of this work is to enable safety analysis for systems that are too high-dimensional for traditional approaches. In this

section we will explore how the proposed method fares in two high-dimensional systems from the OpenAI Gym environment collection [235].

Temporal difference: lunar lander

We first consider a lunar lander system with 6 states $x = [\mathrm{x}, \mathrm{y}, \theta, \dot{\mathrm{x}}, \dot{\mathrm{y}}, \dot{\theta}]$ (vehicle pose and velocities). The signed distance safety function is defined as

$$\begin{aligned} g(x) &= \max\{g_{\text{fly}}(x), g_{\text{land}}(x)\} \ , \\ g_{\text{fly}}(x) &= \min\{\mathrm{x} - \mathrm{x}^w_{\min}, \mathrm{x}^w_{\max} - \mathrm{x},\ \mathrm{y} - \mathrm{y}^w_{\min}, \mathrm{y}^w_{\max} - \mathrm{y}\} \ , \\ g_{\text{land}}(x) &= \min\{\mathrm{x} - \mathrm{x}^p_{\min}, \mathrm{x}^p_{\max} - \mathrm{x},\ \theta - \theta_{\min}, \theta_{\max} - \theta, \dot{\mathrm{y}} - \dot{\mathrm{y}}_{\min}\} \ , \end{aligned} \tag{9.11}$$

Terms marked with superscript w indicate viewing window limits, and terms marked with superscript p indicate landing pad limits. The margin $g(\cdot)$ is thus constructed to allow either flying in free space or landing on the pad; this example illustrates the ability to encode arbitrary state constraints through a signed distance function.

We train 100 Safety DQNs with different random seeds and compare learned values against the observed safety by performing on-policy rollouts in simulation (Figure 9.2). Since computing the safety value function through dynamic programming is intractable on 6-dimensional systems, there is no known ground truth to compare against (Figure 9.3). While the learned Q-value function may be suboptimal, it does give accurate safety predictions for its induced best-effort policy. We present x-y slices of a sample trained value function in Figure 9.5, where the learned safety structure can be seen.

Policy optimization: half-cheetah

Many successful modern reinforcement learning methods use neural networks to directly represent control policies and search for efficient strategies. A number of policy gradient algorithms derive their policy update from the REINFORCE rule [230]:

$$\nabla_\theta \mathbb{E}_{\mathbf{x}\sim\pi_\theta}[\mathcal{V}(\mathbf{x})] = \mathbb{E}_{\mathbf{x}\sim\pi_\theta}[\nabla_\theta \log(p_{\pi_\theta}(\mathbf{x}))\mathcal{V}(\mathbf{x})] \ , \tag{9.12}$$

with $p_{\pi_\theta}(\cdot)$ denoting the probability of taking a trajectory $\mathbf{x}$ under the stochastic policy π_θ parametrized by θ, and $\mathcal{V}(\cdot)$ denoting the outcome of $\mathbf{x}$. Taking $\mathcal{V}(\cdot)$ to represent the time-discounted minimum payoff $g(\cdot)$ of the trajectory as in (9.8), we can directly optimize a policy for discounted safety.

We consider an 18-dimensional half-cheetah system within the MuJoCo physics simulator [237], and define $g(\cdot)$ to be the minimum height of the head and the front leg, so that a failure occurs if either touches the ground (Figure 9.6). Note that we must (at least in part) initialize trajectories at configurations from which the system could in principle maintain safety. Running policy gradient using REINFORCE, all policies trained for discounted safety attempt to balance, though not always successfully, and some learn to sit. In contrast, policies trained with the standard reinforcement learning formulation using $g(\cdot)$ as an

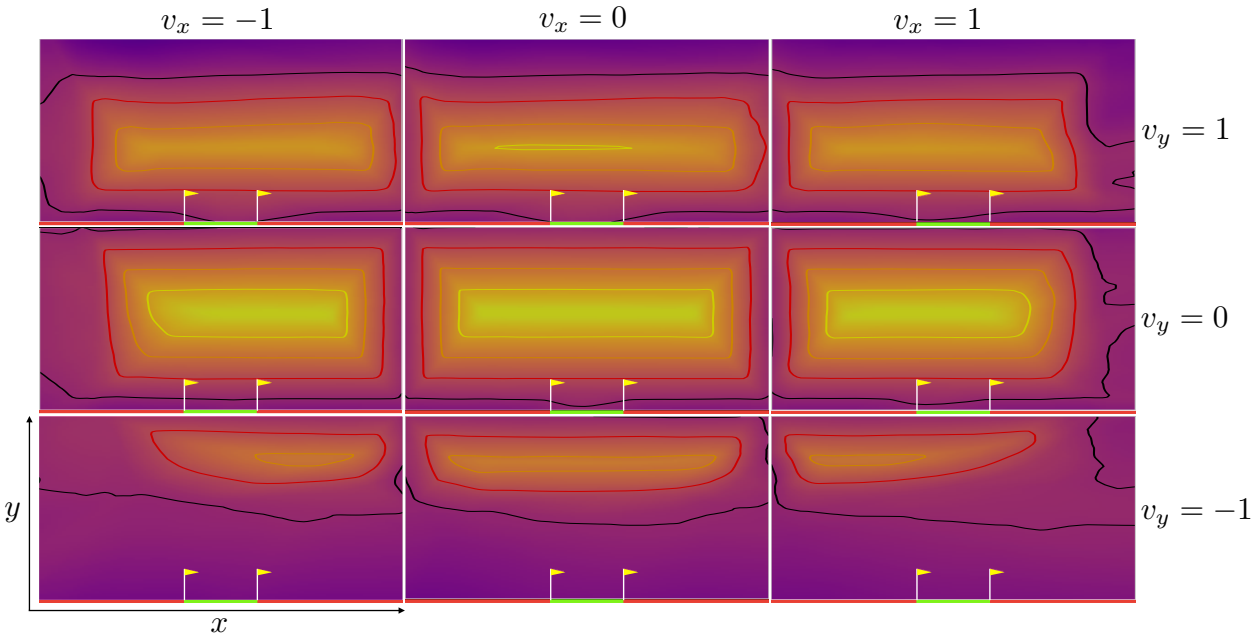


Figure 9.5: Slices of the learned lunar lander value function under Safety Q-Learning, overlaid on the image of the viewing window for $\theta = 0$ and $\dot{\theta} = 0$. Computed safe set boundary in black. At low speeds, the values near the ground are higher close to the landing pad, revealing the effect of g_{land}. For large downward velocities, ground collision is inevitable from the lower half of the screen.

additive reward tend to raise the front leg and sometimes jump, and invariably fall over. Defining an alternative reward that purely penalizes forbidden contacts similarly failed to yield safe learned behaviors.

Using the more sophisticated soft actor-critic (SAC) algorithm [231] we find that after hyper-parameter optimization, all policies trained across 20 random seeds using a discounted sum of $g(\cdot)$ launch the cheetah into the air and always fall over. Using a discounted sum of contact penalties, 65% of policies do learn to sit; however, the remaining 35% produce unsafe jumping behavior. We speculate that the sparsity of the reward signal makes learning challenging. Across the 20 random seeds, all policies trained with discounted safety visibly attempt to stand: 80% of them succeed in doing so reliably, with an additional 5% reliably sitting if standing fails. The different emergent policies are depicted in Figure 9.6.

9.4 Chapter Summary

This chapter has introduced a time-discounted *Safety Bellman Equation* whose unique fixed-point solution converges to the undiscounted Hamilton-Jacobi safety value function as the time discounting is asymptotically relaxed. Our new formulation can readily be used with a wide variety of state-of-the-art reinforcement learning algorithms by a simple modification

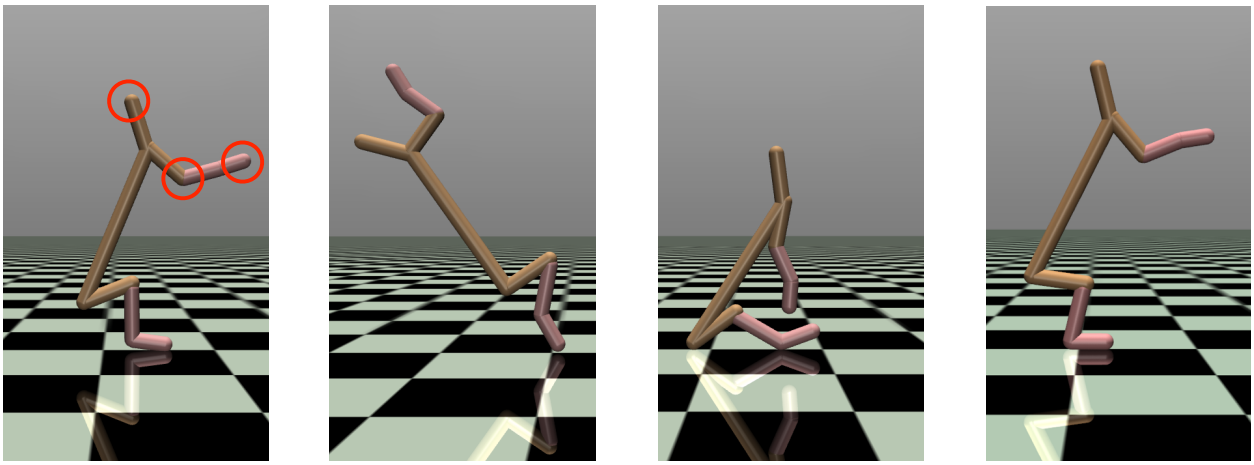


Figure 9.6: Learned half-cheetah safety policies, aimed to keep the head and front leg off the ground, using deep policy optimization methods. *Left to right:* typical starting configuration; an unsafe jumping policy learned using a sum of discounted heights; a safe sitting policy learned using discounted safety or (less reliably) discounted sum of contact penalties; a safe standing policy learned using discounted safety.

of their Bellman update step. We prove the convergence of the resulting *Safety Q-learning* scheme in finite Markov decision processes, which to our knowledge is the first model-free method for computing the safety value function beyond naive Monte Carlo trajectory shooting. Adopting function approximation techniques from modern reinforcement learning, we demonstrate scalability for higher dimensional systems and find that our DQN-based approach is accurate compared to analytic solutions and existing numerical methods. Finally, by directly optimizing the control policy for discounted safety in a system with 18 continuous state dimensions, leading to *Safety REINFORCE* and *Safety SAC*, we find that it is possible to learn control policies that preserve safety with significantly more success than those obtained from the standard reinforcement learning formulation using a sum of discounted rewards.

9.4.1 Implications for Hamilton-Jacobi safety analysis

The contribution of this chapter can be seen as unlocking a family of tools for safety analysis that can be successfully applied to systems intractable for traditional techniques. Once computed, the learned safety analysis can have practical applications in safe robot control and safe reinforcement learning, analogous to existing safety analysis tools. To this end, we expect that robust formulations, as well as research in transfer learning will facilitate the use of simulation-based safety analysis in physical systems; we also note that, subject to model fidelity, conservative approximations of the safety value can always be guaranteed through forward simulation of the computed safety policy [238]. While our focus in this chapter is on

model-free reinforcement learning algorithms, model-based methods such as value and policy iteration can also be employed for safety analysis under the discounted safety formulation. Some preliminary exploration of such methods is done in [232].

9.4.2 Implications for reinforcement learning

By introducing a Safety Bellman Equation that is readily compatible with the reinforcement learning framework, we hope to enable and inspire researchers and practitioners in the field of reinforcement learning to explicitly include safety in their learning algorithms. Ultimately, we hope that, by enabling learning systems to reason about constraint satisfaction, future advances in the field will bring about highly capable intelligent systems that can be deployed safely [195].

9.4.3 Limitations and future work

To reach convergence under model-free learning schemes, the system must repeatedly violate the constraints. Thus model-free algorithms using discounted safety must be used in simulation or an environment where leaving the constraint set does not result in catastrophic failure. It is also important to note that policies obtained with the formulation introduced here will seek to maintain safety but not accomplish another task while staying safe. Effectively combining the discounted safety approach with performance-driven learning is therefore a natural research direction. For example, this can be done by using the learned safety policy in the supervisory control framework introduced in Chapter 6. Since the Safety Q-learning scheme is off-policy, safety analysis can be continually updated in the background even while a system is controlled by a different policy, yielding a natural method for updating safety analysis without the explicit need for other mechanisms such as a Gaussian process. Finally, while deep neural networks are expressive function approximators, their training methods do not in general have convergence guarantees. It may prove fruitful to investigate combining the line of work introduced in this chapter with recent research in neural network verification [9] to provide eventual formal guarantees about learned value functions and control policies.

Chapter 10

Towards a Safe Robotic Future

> Prediction is very difficult,
> especially about the future.
>
> Niels Bohr (1885–1962)
> Physicist

Even as our analytic and computational tools for safety assurance continue to scale and improve in future years, maintaining safety in complex environments will remain a fundamental challenge, because unexpected changes in the world, or the unexpected actions of human agents, may always invalidate design-time guarantees. Our future autonomous systems will need to actively reason about safety assurance online, incorporating observations about the environment and other agents through state-of-the-art statistical learning and intelligently adapting their behavior to ensure safety beyond the reach of offline analysis.

Intractability: Completeness vs. Soundness

As we have seen, the exact computation of optimal safety policies becomes intractable for complex dynamical system models. However, it is often the case that we can compute tractable control policies that can ensure the safety of our system in a wide range of conditions. These are generally suboptimal best-effort solutions based on function approximation (as in Chapter 9) or imposed structure (as in Chapter 4), yet they can be used in conjunction with robust optimal control to provide rigorous and *sound* safety guarantees. This is an important consideration: while *completeness* may not be possible in complex systems (that is, there may be conditions in which a safe control strategy is possible but cannot be tractably computed), *soundness* is enough to ensure the preservation of safety, since the system can limit its actions to those for which it can guarantee safety (following a recursive feasibility criterion along the lines of Chapter 5). This means that, under the appropriate schemes, the price of intractability may be conservativeness, but not loss of safety.

Extracting the Structure

Machine learning enables us to automatically identify problem structure through data analysis. Rather than used in isolation, this capability can be viewed as a useful piece of machinery in the design of autonomous systems, making it possible to perform computational analysis more tractably, as exemplified in Chapter 9. This can be done with the desirable assurances as long as we bear in mind the reality gap between model and system, both during the design process and in the decision-making of our autonomous systems (as in Chapters 6 and 7).

Closing the Learning Loop

The long-term objective of control theory should be to provide strong theoretical guarantees around systems that learn and adapt to observed data, in a way analogous to the guarantees that exist today around feedback control mapping sensor data to actuation decisions.

The challenge of "closing the loop" around sophisticated learning-enabled components to prove correctness properties in advanced automation systems is a pressing priority, as has been recognized by DARPAs new Assured Autonomy project. It also poses a major opportunity for advancing the fields of robotics and control theory. The difficulty in guaranteeing deployment-time safety through offline analysis mirrors the inadequacies of early open-loop analysis for designing stable control systems. Much like the need for stability assurance fueled the development of closed-loop analysis and modern control theory in the 20th century, our understanding of the interplay between system resilience and learning may well be propelled forward this century by the need to ensure the safe autonomous operation of robotic systems in uncertain and changing environments. This understanding will require research at the intersection of control theory, robotics, and machine learning, with the goal of establishing provable properties for learning loops akin to those existing for feedback loops.

Closing the Human-Robot Loop

The other safety-critical loop that will demand close study in the coming decades is the interplay between these increasingly complex automation systems and the human beings they will interact with. In most current instances of safety-critical autonomous robotic systems, from industrial manipulators to autonomous car prototypes, interaction with humans is primarily viewed as a source of risk. Given the limited understanding of human-machine interaction over time, predictions tend to be based on highly simplified models, and safety assurance requires leaving large error margins. Recent advances in machine learning and the growing availability of computational resources present an opportunity to "close the loop" around human-machine interaction, developing more accurate models and using them to reason about the coupled decisions of the human and the automation. If successful, this effort can turn interaction from liability to asset, establishing humans as valuable allies in the quest for safety. This will require fusing robotics expertise with research in cognitive science, human factors engineering, and game theory.

Safe Advanced AI Systems

Beyond the domain of robotics, the growing capabilities of AI technologies are opening a wide range of new applications for automation, often in the purely digital domain. From personal services to critical infrastructure, correct operation of these systems is extremely important, as is the study of the full human-automation system. The recent social concern surrounding privacy violations, emergent algorithmic bias, and uncontrolled propagation of fake news highlights the urgency of ensuring that digitally deployed AI and machine learning systems operate in a well-understood and reliable manner. Extending the previous results for physical safety to a more general constraint satisfaction problem may enable these systems to explicitly reason about whether their decisions may lead to undesirable outcomes when interfacing with human beings. There are additional important challenges in this domain, ranging from the difficulty of reasoning about highly complex human-automation systems to the evasiveness of well-defined constraints that these systems must satisfy. The latter issue links directly to the problem of value alignment, and requires investigating whether the boundaries of acceptable operation can be correctly inferred by an AI system through interaction with its human users.

Robotic technologies have the potential to improve the quality of human life throughout the globe and have a long-lasting positive impact for many generations to come. In order for these benefits to be reaped, we must equip robotic systems with the ability to actively reason about their own safety and continually monitor the validity of their modeling assumptions to provide meaningful assurances. As our understanding of safe robotics and automation becomes solidified and translated into the development of increasingly competent and reliable systems, we can look forward to a future in which we can trust our technology with the future of our children.

Figure 10.1: A vision of a future with safe robotic systems.

Bibliography

[1] F. Vaussard, J. Fink, V. Bauwens, et al. "Lessons Learned from Robotic Vacuum Cleaners Entering the Home Ecosystem". *Robotics and Autonomous Systems*. Advances in Autonomous Robotics — Selected Extended Papers of the Joint 2012 TAROS Conference and the FIRA RoboWorld Congress, Bristol, UK 62.3 (2014), pp. 376–391.

[2] B. S. Peters, P. R. Armijo, C. Krause, et al. "Review of Emerging Surgical Robotic Technology". *Surgical Endoscopy* 32.4 (2018), pp. 1636–1655.

[3] Federal Aviation Administration. *Unmanned Aircraft Systems*. Tech. rep. (Accessed on 2019/10/16.) 2018.

[4] F. Duarte and C. Ratti. "The Impact of Autonomous Vehicles on Cities: A Review". *Journal of Urban Technology* 25.4 (2018), pp. 3–18.

[5] E. Ackerman and M. Koziol. "In the Air with Zipline's Medical Delivery Drones". *IEEE Spectrum: Technology, Engineering, and Science News* (2019). (Accessed on 2019/11/22.)

[6] T. Prevot, J. Rios, P. Kopardekar, et al. "UAS Traffic Management (UTM) Concept of Operations to Safely Enable Low Altitude Flight Operations". *AIAA Aviation Technology, Integration, and Operations Conference*. Washington, D.C.: American Institute of Aeronautics and Astronautics, 2016.

[7] A. Krizhevsky, I. Sutskever, and G. E. Hinton. "ImageNet Classification with Deep Convolutional Neural Networks". *Advances in Neural Information Processing Systems*. Ed. by F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger. Curran Associates, 2012, pp. 1097–1105.

[8] D. Silver, A. Huang, C. J. Maddison, et al. "Mastering the Game of Go with Deep Neural Networks and Tree Search". *Nature* 529.7587 (2016), pp. 484–489.

[9] C. Liu, T. Arnon, C. Lazarus, et al. "Algorithms for Verifying Deep Neural Networks" (2019). arXiv: `1903.06758 [cs, stat]`.

[10] A. Kurakin, I. Goodfellow, and S. Bengio. "Adversarial Examples in the Physical World". *Artificial Intelligence Safety and Security*. Ed. by R. V. Yampolskiy. 1 edition. Boca Raton: Chapman and Hall/CRC, 2018.

[11] S. Tjahjono. *Aircraft Accident Investigation Report, PT. Lion Mentari Airlines, Boeing 737-8 (MAX); PK-LQP, Tanjung Karawang, West Java*. Aircraft Accident Investigation Report KNKT.18.10.35.04. Republic of Indonesia, Komite Nasional Keselamatan Transportasi, 2018.

[12] Aircraft Accident Investigation Bureau. *Aircraft Accident Investigation Preliminary Report: Ethiopian Airlines Group, B737-8 (MAX) Registered ET-AVJ 28 NM South East of Addis Ababa, Bole International Airport*. Tech. rep. AI-01/19. Ethiopian Ministry of Transport, 2019.

[13] J. F. Fisac, M. Chen, C. J. Tomlin, and S. S. Sastry. "Reach-avoid Problems with Time-varying Dynamics, Targets and Constraints". *ACM International Conference on Hybrid Systems: Computation and Control*. 2015, pp. 11–20.

[14] M. Chen, J. F. Fisac, S. Sastry, and C. J. Tomlin. "Safe Sequential Path Planning of Multi-Vehicle Systems via Double-Obstacle Hamilton-Jacobi-Isaacs Variational Inequality". *IEEE European Control Conference (ECC)*. Linz, Austria, 2015, pp. 3304–3309.

[15] M. Chen, S. Bansal, J. F. Fisac, and C. J. Tomlin. "Robust Sequential Trajectory Planning under Disturbances and Adversarial Intruder". *IEEE Transactions on Control Systems Technology* 27.4 (2019), pp. 1566–1582.

[16] D. Fridovich-Keil*, S. L. Herbert*, J. F. Fisac*, et al. "Planning, Fast and Slow: A Framework for Adaptive Real-Time Safe Trajectory Planning." *IEEE International Conference on Robotics and Automation (ICRA)*. 2018.

[17] D. Fridovich-Keil, J. F. Fisac, and C. J. Tomlin. "Safely Probabilistically Complete Real-Time Planning and Exploration in Unknown Environments". *IEEE International Conference on Robotics and Automation (ICRA)*. 2019, pp. 7470–7476.

[18] J. F. Fisac, A. K. Akametalu, M. N. Zeilinger, et al. "A General Safety Framework for Learning-Based Control in Uncertain Robotic Systems". *IEEE Transactions on Automatic Control* 64.7 (2019), pp. 2737–2752.

[19] J. Fisac, A. Bajcsy, S. Herbert, et al. "Probabilistically Safe Robot Planning with Confidence-Based Human Predictions". *Robotics: Science and Systems XIV*. Vol. 14. 2018.

[20] A. Bajcsy, S. L. Herbert, D. Fridovich-Keil, et al. "A Scalable Framework for Real-Time Multi-Robot, Multi-Human Collision Avoidance". *IEEE International Conference on Robotics and Automation (ICRA)*. 2019, pp. 936–943.

[21] J. F. Fisac, E. Bronstein, E. Stefansson, et al. "Hierarchical Game-Theoretic Planning for Autonomous Vehicles". *IEEE International Conference on Robotics and Automation (ICRA)*. 2019, pp. 9590–9596.

[22] S. Sastry. "Nonlinear Systems: Analysis, Stability and Control". Vol. 10. 1999.

[23] C. Tomlin, J. Lygeros, and S. S. Sastry. "A game theoretic approach to controller design for hybrid systems". *Proceedings of the IEEE* 88.7 (2000).

[24] E. A. Coddington and N. Levinson. "Theory of Ordinary Differential Equations". International Series in Pure and Applied Mathematics. New York: McGraw-Hill, 1955.

[25] N. Wiener. "Cybernetics: or control and communication in the animal and the machine." John Wiley, 1948.

[26] J. Zhang, K. H. Johansson, J. Lygeros, and S. Sastry. "Zeno Hybrid Systems". *International Journal of Robust and Nonlinear Control* 11.5 (2001), pp. 435–451.

[27] J.-P. Aubin, A. Bayen, and P. Saint-Pierre. "Viability Theory: New Directions". Springer, 2011.

[28] M. P. Chapman, J. Lacotte, A. Tamar, et al. "A Risk-Sensitive Finite-Time Reachability Approach for Safety of Stochastic Dynamic Systems". *European Control Conference (ECC)*. 2019.

[29] A. Bajcsy, S. Bansal, E. Bronstein, et al. "An Efficient Reachability-Based Framework for Provably Safe Autonomous Navigation in Unknown Environments". *IEEE Conference on Decision and Control (CDC)*. 2019. arXiv: 1905.00532.

[30] C. E. García, D. M. Prett, and M. Morari. "Model predictive control: Theory and practice—A survey". *Automatica* 25.3 (1989), pp. 335–348.

[31] L. S. Pontryagin, E. Mishchenko, V. Boltyanskii, and R. Gamkrelidze. "The mathematical theory of optimal processes". Wiley, 1962.

[32] R. Bellman. "The theory of dynamic programming". *Bulletin of the American Mathematical Society* 60.6 (1954), pp. 503–515.

[33] M. G. Crandall and P.-L. Lions. "Viscosity solutions of Hamilton-Jacobi equations". *Transactions of the American Mathematical Society* 277.1 (1983).

[34] L. C. Evans and P. E. Souganidis. "Differential games and representation formulas for solutions of Hamilton-Jacobi-Isaacs equations". *Indiana University mathematics journal* 33.5 (1984), pp. 773–797.

[35] M. G. Crandall, L. C. Evans, and P. L. Lions. "Some Properties of Viscosity Solutions of Hamilton-Jacobi Equations". *Transactions of the American Mathematical Society* 282.2 (1984).

[36] R. Isaacs. *Differential Games*. Tech. rep. Santa Monica, California: RAND Corporation, 1954.

[37] R. Isaacs. *Games of Pursuit*. Tech. rep. Santa Monica, California: RAND Corporation, 1951.

[38] P. Varaiya. "On the existence of solutions to a differential game". *SIAM Journal on Control* 5.1 (1967), pp. 153–162.

[39] E. Roxin. "Axiomatic approach in differential games". *Journal of Optimization Theory and Applications* 3.3 (1969), pp. 153–163.

[40] R. J. Elliott and N. J. Kalton. "The Existence of Value in Differential Games of Pursuit and Evasion". *Journal of Differential Equations* 12.3 (1972), pp. 504–523.

[41] T. Basar and G. Olsder. "Dynamic Noncooperative Game Theory". 2nd ed. Classics in Applied Mathematics. Society for Industrial and Applied Mathematics, 1998.

[42] T. Basar and A. Haurie. "Feedback Equilibria in Differential Games with Structural and Modal Uncertainties". *Advances in Large Scale Systems*. JAI Press, 1984, pp. 163–201.

[43] M. Simaan and J. B. Cruz. "On the Stackelberg Strategy in Nonzero-Sum Games". *Journal of Optimization Theory and Applications* 11.5 (1973), pp. 533–555.

[44] A. Abate, M. Prandini, J. Lygeros, and S. Sastry. "Probabilistic Reachability and Safety for Controlled Discrete Time Stochastic Hybrid Systems". *Automatica* March (2008).

[45] M. Bardi and I. C. Dolcetta. "Optimal Control and Viscosity Solutions of Hamilton-Jacobi-Bellman Equations". Modern Birkhäuser Classics. Birkhäuser Basel, 1997.

[46] I. M. Mitchell, S. Kaynama, M. Chen, and M. Oishi. "Safety Preserving Control Synthesis for Sampled Data Systems". *Nonlinear Analysis: Hybrid Systems*. Special Issue Related to IFAC Conference on Analysis and Design of Hybrid Systems (ADHS 12) 10 (2013), pp. 63–82.

[47] I. M. Mitchell and S. Kaynama. "An Improved Algorithm for Robust Safety Analysis of Sampled Data Systems". *Hybrid Systems: Computation and Control (HSCC)*. 1. 2015, pp. 21–30.

[48] E. Barron. "Differential Games with Maximum Cost". *Nonlinear Analysis: Theory, Methods & Applications* (1990), pp. 971–989.

[49] 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 (2005), pp. 947–957.

[50] M. G. Crandall and P.-L. Lions. "Two Approximations of Solutions of Hamilton-Jacobi Equations". *Mathematics of Computation* 43.167 (1984).

[51] S. Osher and C.-W. Shu. "High-Order Essentially Nonoscillatory Schemes for Hamilton-Jacobi Equations". *SIAM Journal on Numerical Analysis* 28.4 (1991), pp. 907–922.

[52] S. Osher and R. Fedkiw. "Level Set Methods and Dynamic Implicit Surfaces". Vol. 153. Applied Mathematical Sciences. New York, NY: Springer New York, 2003.

[53] I. Mitchell and J. Templeton. "A Toolbox of Hamilton-Jacobi Solvers for Analysis of Nondeterministic Continuous and Hybrid Systems". *Hybrid Systems: Computation and Control* (2005), pp. 480–494.

[54] A. B. Kurzhanski and P. Varaiya. "Ellipsoidal Techniques for Reachability Analysis". *Hybrid Systems: Computation and Control*. Ed. by N. Lynch and B. H. Krogh. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2000, pp. 202–214.

[55] J. N. Maidens, S. Kaynama, I. M. Mitchell, et al. "Lagrangian methods for approximating the viability kernel in high-dimensional systems". *Automatica* 49.7 (2013), pp. 2017–2029.

[56] S. Kaynama, I. M. Mitchell, M. Oishi, and G. A. Dumont. "Scalable safety-preserving robust control synthesis for continuous-time linear systems". *IEEE Transactions on Automatic Control* 60.11 (2015), pp. 3065–3070. eprint: 1312.3399.

[57] E. Asarin, T. Dang, and O. Maler. "The d/Dt Tool for Verification of Hybrid Systems". *Computer Aided Verification*. Ed. by E. Brinksma and K. G. Larsen. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2002, pp. 365–370.

[58] C. Le Guernic and A. Girard. "Reachability Analysis of Linear Systems Using Support Functions". *Nonlinear Analysis: Hybrid Systems*. IFAC World Congress 2008 4.2 (2010), pp. 250–262.

[59] P. S. Duggirala, S. Mitra, M. Viswanathan, and M. Potok. "C2E2: A Verification Tool for Stateflow Models". *Tools and Algorithms for the Construction and Analysis of Systems*. Ed. by C. Baier and C. Tinelli. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2015, pp. 68–82.

[60] S. Kong, S. Gao, W. Chen, and E. Clarke. "dReach: δ-Reachability Analysis for Hybrid Systems". *Tools and Algorithms for the Construction and Analysis of Systems*. Ed. by C. Baier and C. Tinelli. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2015, pp. 200–205.

[61] S. Sastry and M. Bodson. "Adaptive Control: Stability, Convergence and Robustness". Mineola, N.Y: Dover Publications, 2011.

[62] R. S. Sutton and A. G. Barto. "Reinforcement Learning: An Introduction". 2nd ed. MIT Press, 2018.

[63] R. Sutton, A. Barto, and R. Williams. "Reinforcement Learning Is Direct Adaptive Optimal Control". *IEEE Control Systems Magazine* 12.2 (1992), pp. 19–22.

[64] S. A. Billings. "Nonlinear System Identification : NARMAX Methods in the Time, Frequency, and Spatio-Temporal Domains". Wiley, 2013.

[65] M. Gevers. "Identification for Control: From the Early Achievements to the Revival of Experiment Design*". *European Journal of Control* 11.4 (2005), pp. 335–352.

[66] R. E. Kalman. "A New Approach to Linear Filtering and Prediction Problems". *Journal of Basic Engineering* 82.1 (1960), pp. 35–45.

[67] N. Gordon, D. Salmond, and A. Smith. "Novel Approach to Nonlinear/Non-Gaussian Bayesian State Estimation". *IEE Proceedings F - Radar and Signal Processing* 140.2 (1993), pp. 107–113.

[68] C. J. C. H. Watkins and P. Dayan. "Q-Learning". *Machine Learning* 8.3 (1992), pp. 279–292.

[69] G. A. Miller. "The Cognitive Revolution: A Historical Perspective". *Trends in Cognitive Sciences* 7.3 (2003), pp. 141–144.

[70] D. Marr and T. Poggio. *From Understanding Computation to Understanding Neural Circuitry*. Artificial Intelligence Laboratory. A.I. Memo. Massachusetts Institute of Technology, 1976.

[71] T. L. Griffiths, F. Lieder, and N. D. Goodman. "Rational Use of Cognitive Resources: Levels of Analysis between the Computational and the Algorithmic". *Topics in Cognitive Science* 7.2 (2015), pp. 217–229.

[72] A. Tversky and D. Kahneman. "Availability: A Heuristic for Judging Frequency and Probability". *Cognitive Psychology* 5.2 (1973), pp. 207–232.

[73] A. Tversky and D. Kahneman. "Judgment under Uncertainty: Heuristics and Biases". *Science* 185.4157 (1974), pp. 1124–1131.

[74] R. D. Luce. "Individual choice behavior: A theoretical analysis". Wiley, 1959.

[75] R. E. Kalman. "When Is a Linear Control System Optimal?" *Journal of Basic Engineering* 86.1 (1964), pp. 51–60.

[76] A. Y. Ng and S. J. Russell. "Algorithms for Inverse Reinforcement Learning". *ACM International Conference on Machine Learning (ICML)*. San Francisco, CA, USA: Morgan Kaufmann Publishers, 2000, pp. 663–670.

[77] N. D. Ratliff, J. A. Bagnell, and M. A. Zinkevich. "Maximum Margin Planning". *ACM International Conference on Machine Learning (ICML)*. New York, NY, USA, 2006, pp. 729–736.

[78] D. Ramachandran and E. Amir. "Bayesian Inverse Reinforcement Learning". *International Joint Conference on Artificial Intelligence (IJCAI)*. San Francisco, CA, USA: Morgan Kaufmann Publishers, 2007, pp. 2586–2591.

[79] B. D. Ziebart, A. Maas, J. A. Bagnell, and A. K. Dey. "Maximum Entropy Inverse Reinforcement Learning". *AAAI Conference on Artificial Intelligence*. 2008.

[80] H. Wimmer and J. Perner. "Beliefs about Beliefs: Representation and Constraining Function of Wrong Beliefs in Young Children's Understanding of Deception". *Cognition* 13.1 (1983), pp. 103–128.

[81] A. Gopnik and H. M. Wellman. "Why the Child's Theory of Mind Really Is a Theory". *Mind & Language* 7.1-2 (1992), pp. 145–171.

[82] A. N. Meltzoff. "Understanding the Intentions of Others: Re-Enactment of Intended Acts by 18-Month-Old Children". *Developmental psychology* 31.5 (1995), pp. 838–850.

[83] A. Dragan and S. Srinivasa. "Generating Legible Motion". *Robotics: Science and Systems*. Vol. 09. 2013.

[84] J. F. Fisac, C. Liu, J. B. Hamrick, et al. "Generating Plans That Predict Themselves". *Workshop on the Algorithmic Foundations of Robotics (WAFR)*. 2016.

[85] E. Barron and H. Ishii. "The Bellman equation for minimizing the maximum cost". *Nonlinear Analysis: Theory, Methods & Applications* (1989).

[86] J. A. Sethian. "A Fast Marching Level Set Method for Monotonically Advancing Fronts". *Proceedings of the National Academy of Sciences* 93.4 (1996), pp. 1591–1595.

[87] A. E. Rapaport. "Characterization of Barriers of Differential Games". *Journal of optimization theory and applications* 97.I (1998), pp. 151–179.

[88] O. Bokanowski, N. Forcadel, and H. Zidani. "Reachability and Minimal Times for State Constrained Nonlinear Problems without Any Controllability Assumption". *SIAM Journal on Control and Optimization* 48.7 (2010), pp. 4292–4316.

[89] O. Bokanowski and H. Zidani. "Minimal time problems with moving targets and obstacles". *18th IFAC World Congress* (2011).

[90] P. Cardaliaguet. "A double obstacle problem arising in differential game theory". *Journal of Mathematical Analysis and Applications* 360.1 (2009), pp. 95–107.

[91] A. Cosso. "Stochastic Differential Games Involving Impulse Controls and Double-Obstacle Quasi-Variational Inequalities". *SIAM Journal on Control and Optimization* 51.3 (2013), pp. 2102–2131.

[92] I. J. Fialho and T. T. Georgiou. "Worst case analysis of nonlinear systems". *IEEE Transactions on Automatic Control* 44.6 (1999), pp. 1180–1196.

[93] K. Margellos and J. Lygeros. "Hamilton-Jacobi Formulation for Reach-Avoid Differential Games". *IEEE Transactions on Automatic Control* 56.8 (2011), pp. 1849–1861.

[94] E. Prassler, A. Ritter, C. Schaeffer, and P. Fiorini. "A Short History of Cleaning Robots". *Autonomous Robots* 9.3 (2000), pp. 211–226.

[95] M. R. Kirchner, R. Mar, G. Hewer, et al. "Time-Optimal Collaborative Guidance Using the Generalized Hopf Formula". *IEEE Control Systems Letters* 2.2 (2018), pp. 201–206.

[96] Federal Aviation Administration. "Automatic Dependent Surveillance-Broadcast (ADS-B) out Performance Requirements to Support Air Traffic Control (ATC) Service; Final Rule". *Federal Register—Rules and Regulations*. 14 CFR Part 91 75.103 (2010), pp. 30159–30195.

[97] Federal Aviation Administration. "UAS Sightings Report". (Accessed on 2019/10/16.) 2019.

[98] Unmanned Vehicle Systems International, Academy of Model Aeronautics, and Federal Aviation Administration. "Know before You Fly". (Accessed on 2019/10/16.) 2019.

[99] Federal Aviation Administration. "National Drone Safety Awareness Week". (Accessed on 2019/10/16.) 2019.

[100] F. Manjoo. "The Autonomous Selfie Drone Is Here. Is Society Ready for It?" *The New York Times* (2018). (Accessed on 2019/10/16.)

[101] S. Hollister. "Skydio 2: The Self-Flying Future of Drones Starts at $999". (Accessed on 2019/10/16.) 2019.

[102] Google Inc. *Google UAS Airspace System Overview*. Tech. rep. 2015.

[103] Amazon.com Inc. *Revising the Airspace Model for the Safe Integration of Small Unmanned Aircraft Systems Airspace Design*. Tech. rep. 2015.

[104] B. Breen. "Controlled Flight Into Terrain and the Enhanced Ground Proximity Warning System". *IEEE Aerospace and Electronic Systems Magazine* 14.1 (1999), pp. 19–24.

[105] Federal Aviation Administration. "Introduction to TCAS II (Version 7.1)". United States Department of Transportation, 2011.

[106] I. Hwang and C. J. Tomlin. "Protocol-Based Conflict Resolution for Finite Information Horizon". *American Control Conference (ACC)*. Vol. 1. 2002, pp. 748–753.

[107] E. Rimon and D. Koditschek. "Exact Robot Navigation Using Artificial Potential Functions". *IEEE Transactions on Robotics and Automation* 8.5 (1992), pp. 501–518.

[108] R. Olfati-Saber and R. M. Murray. "Distributed Cooperative Control of Multiple Vehicle Formations Using Structural Potential Functions". *IFAC Proceedings Volumes*. 15th IFAC World Congress 35.1 (2002), pp. 495–500.

[109] D. E. Chang, S. C. Shadden, J. E. Marsden, and R. Olfati-Saber. "Collision Avoidance for Multiple Agent Systems". *IEEE Conference on Decision and Control (CDC)*. 2003, pp. 539–543.

[110] P. Fiorini and Z. Shiller. "Motion Planning in Dynamic Environments Using Velocity Obstacles". *The International Journal of Robotics Research* 17.7 (1998), pp. 760–772.

[111] E. Frazzoli, Z.-H. Mao, J.-H. Oh, and E. Feron. "Resolution of Conflicts Involving Many Aircraft via Semidefinite Programming". *Journal of Guidance, Control, and Dynamics* 24.1 (2001), pp. 79–86.

[112] J. van den Berg, M. Lin, and D. Manocha. "Reciprocal Velocity Obstacles for Real-Time Multi-Agent Navigation". *IEEE International Conference on Robotics and Automation (ICRA)*. 2008, pp. 1928–1935.

[113] E. Lalish and K. A. Morgansen. "Distributed Reactive Collision Avoidance". *Autonomous Robots* 32.3 (2012), pp. 207–226.

[114] A. Giese, D. Latypov, and N. M. Amato. "Reciprocally-Rotating Velocity Obstacles". *IEEE International Conference on Robotics and Automation (ICRA)*. 2014, pp. 3234–3241.

[115] P. E. Hart, N. J. Nilsson, and B. Raphael. "A Formal Basis for the Heuristic Determination of Minimum Cost Paths". *IEEE Transactions on Systems Science and Cybernetics* 4.2 (1968), pp. 100–107.

[116] S. Karaman and E. Frazzoli. "Sampling-Based Algorithms for Optimal Motion Planning". *The International Journal of Robotics Research* 30.7 (2011), pp. 846–894.

[117] M. Erdmann and T. Lozano-Pérez. "On Multiple Moving Objects". *Algorithmica* 2.1 (1987).

[118] J. van den Berg and M. Overmars. "Prioritized Motion Planning for Multiple Robots". *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2005, pp. 430–435.

[119] M. Chen, S. Bansal, K. Tanabe, and C. J. Tomlin. "Provably Safe and Robust Drone Routing via Sequential Path Planning: A Case Study in San Francisco and the Bay Area" (2017). arXiv: `1705.04585 [cs]`.

[120] J. Lygeros, C. Tomlin, and S. Sastry. "Multiobjective Hybrid Controller Synthesis". *Hybrid and Real-Time Systems*. Ed. by O. Maler. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 1997, pp. 109–123.

[121] S. M. Lavalle. *Rapidly-Exploring Random Trees: A New Tool for Path Planning*. Tech. rep. TR 98–11. Iowa State University, 1998.

[122] S. M. LaValle and J. J. Kuffner. "Randomized Kinodynamic Planning". *The International Journal of Robotics Research* 20.5 (2001), pp. 378–400.

[123] L. E. Kavraki, P. Svestka, J.-. Latombe, and M. H. Overmars. "Probabilistic Roadmaps for Path Planning in High-Dimensional Configuration Spaces". *IEEE Transactions on Robotics and Automation* 12.4 (1996), pp. 566–580.

[124] A. Majumdar and R. Tedrake. "Funnel Libraries for Real-Time Robust Feedback Motion Planning". *The International Journal of Robotics Research* 36.8 (2017), pp. 947–982.

[125] S. Singh, A. Majumdar, J.-J. Slotine, and M. Pavone. "Robust Online Motion Planning via Contraction Theory and Convex Optimization". *2017 IEEE International Conference on Robotics and Automation (ICRA)*. 2017, pp. 5883–5890.

[126] A. Richards and J. How. "Model Predictive Control of Vehicle Maneuvers with Guaranteed Completion Time and Robust Feasibility". *Proceedings of the 2003 American Control Conference, 2003.* Vol. 5. 2003, pp. 4034–4040.

[127] U. Rosolia and F. Borrelli. "Learning Model Predictive Control for Iterative Tasks. a Data-Driven Control Framework". *IEEE Transactions on Automatic Control* 63.7 (2018), pp. 1883–1896.

[128] T. Schouwenaars, J. How, and E. Feron. "Decentralized Cooperative Trajectory Planning of Multiple Aircraft with Hard Safety Guarantees". *AIAA Guidance, Navigation, and Control Conference and Exhibit*. Providence, Rhode Island, 2004.

[129] B. Yamauchi. "A Frontier-Based Approach for Autonomous Exploration". *IEEE International Symposium on Computational Intelligence in Robotics and Automation (CIRA)*. 1997, pp. 146–151.

[130] B. Yamauchi. "Frontier-Based Exploration Using Multiple Robots". *ACM International Conference on Autonomous Agents*. New York, NY, USA, 1998, pp. 47–53.

[131] L. Yoder and S. Scherer. "Autonomous Exploration for Infrastructure Modeling with a Micro Aerial Vehicle". *Field and Service Robotics: Results of the 10th International Conference*. Ed. by D. S. Wettergreen and T. D. Barfoot. Springer Tracts in Advanced Robotics. Cham: Springer, 2016, pp. 427–440.

[132] A. Stentz. "Optimal and Efficient Path Planning for Partially-Known Environments". *Proceedings of the 1994 IEEE International Conference on Robotics and Automation*. 1994, pp. 3310–3317.

[133] S. Koenig and M. Likhachev. "Fast Replanning for Navigation in Unknown Terrain". *IEEE Transactions on Robotics* 21.3 (2005), pp. 354–363.

[134] K. E. Bekris and L. E. Kavraki. "Greedy but Safe Replanning under Kinodynamic Constraints". *IEEE International Conference on Robotics and Automation (ICRA)*. 2007, pp. 704–710.

[135] T. Fraichard and H. Asama. "Inevitable Collision States — a Step towards Safer Robots?" *Advanced Robotics* 18.10 (2004), pp. 1001–1024.

[136] L. Janson, T. Hu, and M. Pavone. "Safe Motion Planning in Unknown Environments: Optimality Benchmarks and Tractable Policies". *Robotics: Science and Systems*. Vol. 14. 2018.

[137] T. M. Moldovan and P. Abbeel. "Safe Exploration in Markov Decision Processes". *ACM International Conference on Machine Learning (ICML)*. USA: Omnipress, 2012, pp. 1451–1458.

[138] J. Achiam, D. Held, A. Tamar, and P. Abbeel. "Constrained Policy Optimization". *ACM International Conference on Machine Learning (ICML)*. 2017, pp. 22–31.

[139] F. Berkenkamp, M. Turchetta, A. P. Schoellig, and A. Krause. "Safe Model-Based Reinforcement Learning with Stability Guarantees". *Advances in Neural Information Processing Systems*. USA: Curran Associates, 2017, pp. 908–919.

[140] Y. Chow, O. Nachum, E. Duenez-Guzman, and M. Ghavamzadeh. "A Lyapunov-Based Approach to Safe Reinforcement Learning". *Advances in Neural Information Processing Systems 31*. Ed. by S. Bengio, H. Wallach, H. Larochelle, et al. Curran Associates, 2018, pp. 8092–8101.

[141] S. L. Herbert, M. Chen, S. Han, et al. "FaSTrack: A Modular Framework for Fast and Guaranteed Safe Motion Planning". *IEEE Conference on Decision and Control (CDC)*. 2017, pp. 1517–1522.

[142] M. Quigley, B. Gerkey, K. Conley, et al. "ROS: An Open-Source Robot Operating System". *ICRA Workshop on Open Source Software*. 2009.

[143] I. A. Sucan, M. Moll, and L. E. Kavraki. "The Open Motion Planning Library". *IEEE Robotics Automation Magazine* 19.4 (2012), pp. 72–82.

[144] J. D. Gammell, S. S. Srinivasa, and T. D. Barfoot. "Batch Informed Trees (BIT*): Sampling-Based Optimal Planning via the Heuristically Guided Search of Implicit Random Geometric Graphs". *IEEE International Conference on Robotics and Automation (ICRA)*. 2015, pp. 3067–3074.

[145] M. Chen, S. L. Herbert, M. S. Vashishtha, et al. "Decomposition of Reachable Sets and Tubes for a Class of Nonlinear Systems". *IEEE Transactions on Automatic Control* 63.11 (2018), pp. 3675–3688.

[146] S. Karaman and E. Frazzoli. "Sampling-Based Optimal Motion Planning for Non-Holonomic Dynamical Systems". *IEEE International Conference on Robotics and Automation (ICRA)*. 2013, pp. 5041–5047.

[147] I. A. Şucan and L. E. Kavraki. "Kinodynamic Motion Planning by Interior-Exterior Cell Exploration". *Algorithmic Foundation of Robotics VIII: Selected Contributions of the Eight International Workshop on the Algorithmic Foundations of Robotics*. Ed. by G. S. Chirikjian, H. Choset, M. Morales, and T. Murphey. Springer Tracts in Advanced Robotics. Berlin, Heidelberg: Springer, 2010, pp. 449–464.

[148] V. Mnih, K. Kavukcuoglu, D. Silver, et al. "Human-Level Control through Deep Reinforcement Learning". *Nature* 518.7540 (2015), pp. 529–533.

[149] J. Schulman, S. Levine, P. Abbeel, et al. "Trust Region Policy Optimization". *ACM International Conference on Machine Learning (ICML)*. 2015, pp. 1889–1897.

[150] P. Abbeel, A. Coates, M. Quigley, and A. Y. Ng. "An Application of Reinforcement Learning to Aerobatic Helicopter Flight". *Advances in Neural Information Processing Systems 19*. Ed. by B. Schölkopf, J. C. Platt, and T. Hoffman. MIT Press, 2007.

[151] A. Coates, P. Abbeel, and A. Y. Ng. "Learning for Control from Multiple Demonstrations". *ACM International Conference on Machine Learning (ICML)*. New York, NY, USA, 2008, pp. 144–151.

[152] J. Z. Kolter, C. Plagemann, D. T. Jackson, et al. "A Probabilistic Approach to Mixed Open-Loop and Closed-Loop Control, with Application to Extreme Autonomous Driving". *IEEE International Conference on Robotics and Automation (ICRA)*. 2010, pp. 839–845.

[153] S. Lupashin, A. Schöllig, M. Sherback, and R. D'Andrea. "A Simple Learning Strategy for High-Speed Quadrocopter Multi-Flips". *IEEE International Conference on Robotics and Automation (ICRA)*. 2010, pp. 1642–1648.

[154] A. Hobbs. "Unmanned Aircraft Systems". *Human Factors in Aviation (Second Edition)*. Ed. by E. Salas and D. Maurino. San Diego: Academic Press, 2010, pp. 505–531.

[155] P. Christiano, Z. Shah, I. Mordatch, et al. "Transfer from Simulation to Real World through Learning Deep Inverse Dynamics Model" (2016). arXiv: `1610.03518 [cs]`.

[156] S. Huang, N. Papernot, I. Goodfellow, et al. "Adversarial Attacks on Neural Network Policies" (2017). arXiv: `1702.02284 [cs, stat]`.

[157] T. J. Perkins and A. G. Barto. "Lyapunov Design for Safe Reinforcement Learning". *Journal of Machine Learning Research* 3.Dec (2002), pp. 803–832.

[158] J. W. Roberts, I. R. Manchester, and R. Tedrake. "Feedback Controller Parameterizations for Reinforcement Learning". *IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning (ADPRL)*. 2011, pp. 310–317.

[159] P. Geibel and F. Wysotzki. "Risk-Sensitive Reinforcement Learning Applied to Control under Constraints". *Journal of Artificial Intelligence Research* 24.1 (2005), pp. 81–108.

[160] J. García and F. Fernández. "A Comprehensive Survey on Safe Reinforcement Learning". *Journal of Machine Learning Research* 16.42 (2015), pp. 1437–1480.

[161] F. Berkenkamp, R. Moriconi, A. P. Schoellig, and A. Krause. "Safe Learning of Regions of Attraction for Uncertain, Nonlinear Systems with Gaussian Processes". *IEEE Conference on Decision and Control (CDC)*. 2016, pp. 4661–4666.

[162] A. Aswani, H. Gonzalez, S. S. Sastry, and C. Tomlin. "Provably Safe and Robust Learning-Based Model Predictive Control". *Automatica (Journal of IFAC)* 49.5 (2013), pp. 1216–1226.

[163] S. Prajna and A. Jadbabaie. "Safety Verification of Hybrid Systems Using Barrier Certificates". *Hybrid Systems: Computation and Control*. Ed. by R. Alur and G. J. Pappas. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2004, pp. 477–492.

[164] C. Sloth, G. J. Pappas, and R. Wisniewski. "Compositional Safety Analysis Using Barrier Certificates". *ACM International Conference on Hybrid Systems: Computation and Control*. New York, NY, USA, 2012, pp. 15–24.

[165] A. D. Ames, J. W. Grizzle, and P. Tabuada. "Control Barrier Function Based Quadratic Programs with Application to Adaptive Cruise Control". *IEEE Conference on Decision and Control (CDC)*. 2014, pp. 6271–6278.

[166] J. H. Gillula and C. J. Tomlin. "Guaranteed Safe Online Learning via Reachability: Tracking a Ground Target Using a Quadrotor". *IEEE International Conference on Robotics and Automation (ICRA)*. 2012, pp. 2723–2730.

[167] J. Gillula and C. Tomlin. "Reducing Conservativeness in Safety Guarantees by Learning Disturbances Online: Iterated Guaranteed Safe Online Learning". *Robotics: Science and Systems*. Vol. 08. 2012.

[168] S. Kaynama and M. Oishi. "A Modified Riccati Transformation for Decentralized Computation of the Viability Kernel under LTI Dynamics". *IEEE Transactions on Automatic Control* 58.11 (2013), pp. 2878–2892.

[169] M. Chen, S. Herbert, and C. J. Tomlin. "Fast Reachable Set Approximations via State Decoupling Disturbances". *IEEE Conference on Decision and Control (CDC)*. 2016, pp. 191–196.

[170] J. Darbon and S. Osher. "Algorithms for overcoming the curse of dimensionality for certain Hamilton-Jacobi equations arising in control theory and elsewhere". *Research in the Mathematical Sciences* 3.1 (2016).

[171] J. Z. Kolter and A. Y. Ng. "Policy Search via the Signed Derivative". *Robotics: Science and Systems*. Vol. 05. 2009.

[172] C. E. Rasmussen and C. K. I. Williams. "Gaussian Processes for Machine Learning". Adaptive Computation and Machine Learning. OCLC: ocm61285753. Cambridge, Mass: MIT Press, 2006.

[173] A. Genz. "Numerical Computation of Multivariate Normal Probabilities". *Journal of Computational and Graphical Statistics* 1.2 (1992), pp. 141–149.

[174] D. Fridovich-Keil, A. Bajcsy, J. F. Fisac, et al. "Confidence-Aware Motion Prediction for Real-Time Collision Avoidance". *The International Journal of Robotics Research* (2019).

[175] B. S. Dhillon. "Robot System Reliability and Safety: A Modern Approach". CRC Press, 2015.

[176] J. Vincent. "Mall Security Bot Knocks down Toddler, Breaks Asimov's First Law of Robotics". *The Verge* (2016). (Accessed on 2019/10/20.)

[177] N. Garun. "DC Security Robot Quits Job by Drowning Itself in a Fountain". *The Verge* (2017). (Accessed on 2019/10/20.)

[178] B. D. Ziebart, N. Ratliff, G. Gallagher, et al. "Planning-Based Prediction for Pedestrians". *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2009, pp. 3931–3936.

[179] H. Kretzschmar, M. Spies, C. Sprunk, and W. Burgard. "Socially Compliant Mobile Robot Navigation via Inverse Reinforcement Learning". *The International Journal of Robotics Research* 35.11 (2016), pp. 1289–1307.

[180] C. L. Baker, R. Saxe, and J. B. Tenenbaum. "Action Understanding as Inverse Planning". *Cognition* 113.3 (2009), pp. 329–349.

[181] H. Ben Amor, G. Neumann, S. Kamthe, et al. "Interaction Primitives for Human-Robot Cooperation Tasks". *IEEE International Conference on Robotics and Automation (ICRA)*. 2014, pp. 2831–2837.

[182] H. Ding, G. Reißig, K. Wijaya, et al. "Human Arm Motion Modeling and Long-Term Prediction for Safe and Efficient Human-Robot-Interaction". *IEEE International Conference on Robotics and Automation (ICRA)*. 2011, pp. 5875–5880.

[183] H. S. Koppula and A. Saxena. "Anticipating human activities for reactive robotic response." *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2013, p. 2071.

[184] P. A. Lasota and J. A. Shah. "Analyzing the effects of human-aware motion planning on close-proximity human–robot collaboration". *Human factors* 57.1 (2015), pp. 21–33.

[185] K. P. Hawkins, N. Vo, S. Bansal, and A. F. Bobick. "Probabilistic Human Action Prediction and Wait-Sensitive Planning for Responsive Human-Robot Collaboration". *International Conference on Humanoid Robots (Humanoids)*. IEEE-RAS, 2013, pp. 499–506.

[186] E. Schmerling, K. Leung, W. Vollprecht, and M. Pavone. "Multimodal Probabilistic Model-Based Planning for Human-Robot Interaction". *IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018.

[187] G. S. Aoude, B. D. Luders, J. M. Joseph, et al. "Probabilistically Safe Motion Planning to Avoid Dynamic Obstacles with Uncertain Motion Patterns". *Auton. Robots* 35.1 (2013), pp. 51–76.

[188] J. Von Neumann and O. Morgenstern. "Theory of games and economic behavior". Princeton University Press, 1945.

[189] C. Finn, S. Levine, and P. Abbeel. "Guided cost learning: Deep inverse optimal control via policy optimization". *ACM International Conference on Machine Learning (ICML)*. 2016, pp. 49–58.

[190] D. Ridel, E. Rehder, M. Lauer, et al. "A Literature Review on the Prediction of Pedestrian Behavior in Urban Scenarios". *IEEE International Conference on Intelligent Transportation Systems (ITSC)*. 2018, pp. 3105–3112.

[191] D. Helbing and P. Molnár. "Social Force Model for Pedestrian Dynamics". *Physical Review E* 51.5 (1995), pp. 4282–4286.

[192] V. Karasev, A. Ayvaci, B. Heisele, and S. Soatto. "Intent-Aware Long-Term Prediction of Pedestrian Motion". *IEEE International Conference on Robotics and Automation (ICRA)*. 2016, pp. 2543–2549.

[193] W.-C. Ma, D.-A. Huang, N. Lee, and K. M. Kitani. "Forecasting Interactive Dynamics of Pedestrians with Fictitious Play". *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017, pp. 774–782.

[194] N. Wiener. "Some Moral and Technical Consequences of Automation". *Science* 131.3410 (1960), pp. 1355–1358.

[195] D. Amodei, C. Olah, J. Steinhardt, et al. "Concrete Problems in Ai Safety" (2016). arXiv: `1606.06565 [cs]`.

[196] D. Hadfield-Menell, S. J. Russell, P. Abbeel, and A. Dragan. "Cooperative Inverse Reinforcement Learning". *Advances in Neural Information Processing Systems*. Ed. by D. D. Lee, M. Sugiyama, U. V. Luxburg, et al. Curran Associates, 2016.

[197] A. Bajcsy, D. P. Losey, M. K. O'Malley, and A. D. Dragan. "Learning Robot Objectives from Physical Human Interaction". *Conference on Robot Learning*. 2017, pp. 217–226.

[198] A. Bobu, A. Bajcsy, J. F. Fisac, and A. D. Dragan. "Learning under Misspecified Objective Spaces". *Conference on Robot Learning*. 2018, pp. 796–805.

[199] Federal Highway Administration. *Highway Statistics, 2018*. Tech. rep. United States Department of Transportation, 2019.

[200] World Health Organization. *Global Status Report on Road Safety 2018*. Tech. rep. Geneva, 2018.

[201] National Highway Traffic Safety Administration. *Critical Reasons for Crashes Investigated in the National Motor Vehicle Crash Causation Survey*. Tech. rep. DOT HS 812 506. United States Department of Transportation, 2018.

[202] A. Carvalho, G. Palmieri, H. E. Tseng, et al. "Robust Vehicle Stability Control with an Uncertain Driver Model". *European Control Conference (ECC)*. 2013.

[203] M. P. Vitus and C. J. Tomlin. "A Probabilistic Approach to Planning and Control in Autonomous Urban Driving". *IEEE Conference on Decision and Control (CDC)*. 2013, pp. 2459–2464.

[204] B. Luders, M. Kothari, and J. How. "Chance Constrained RRT for Probabilistic Robustness to Environmental Uncertainty". *AIAA Guidance, Navigation, and Control Conference*. 2010.

[205] C. Hermes, C. Wohler, K. Schenk, and F. Kummert. "Long-Term Vehicle Motion Prediction". *IEEE Intelligent Vehicles Symposium*. 2009, pp. 652–657.

[206] P. Trautman and A. Krause. "Unfreezing the Robot: Navigation in Dense, Interacting Crowds". *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2010, pp. 797–803.

[207] Reuters. "Waymo's Self Driving Minivans Struggles to Merge in Left Lane". *Daily Mail* (2018).

[208] T. B. Lee. "Even Self-Driving Leader Waymo Is Struggling to Reach Full Autonomy". *Ars Technica* (2018).

[209] D. Sadigh, S. Sastry, S. A. Seshia, and A. D. Dragan. "Planning for Autonomous Cars That Leverage Effects on Human Actions". *Robotics: Science and Systems*. 2016.

[210] D. Sadigh, S. S. Sastry, S. A. Seshia, and A. Dragan. "Information Gathering Actions over Human Internal State". *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE/RSJ, 2016, pp. 66–73.

[211] National Transportation Safety Board. *Collison between Vehicle Controlled by Developmental Automated Driving System and Pedestrian*. Tech. rep. NTSB/HAR-19/03. Washington, D.C., 2019.

[212] A. Liniger and J. Lygeros. "A Noncooperative Game Approach to Autonomous Racing". *IEEE Transactions on Control Systems Technology* (2019), pp. 1–14.

[213] A. Dreves and M. Gerdts. "A Generalized Nash Equilibrium Approach for Optimal Control Problems of Autonomous Cars". *Optimal Control Applications and Methods* 39.1 (2018), pp. 326–342.

[214] R. Spica, D. Falanga, E. Cristofalo, et al. "A Real-Time Game Theoretic Planner for Autonomous Two-Player Drone Racing". *Robotics: Science and Systems*. Vol. 14. 2018.

[215] T. B. Sheridan. "Three Models of Preview Control". *IEEE Transactions on Human Factors in Electronics* HFE-7.2 (1966), pp. 91–102.

[216] H. Peng and M. Tomizuka. "Preview Control for Vehicle Lateral Guidance in Highway Automation". *Journal of Dynamic Systems, Measurement, and Control* 115.4 (1993), pp. 679–686.

[217] C. C. Macadam. "Understanding and Modeling the Human Driver". *Vehicle System Dynamics* 40.1-3 (2003), pp. 101–134.

[218] K. Waugh, B. Ziebart, and D. Bagnell. "Computational Rationalization: The Inverse Equilibrium Problem". *ACM International Conference on Machine Learning (ICML)*. Ed. by L. Getoor and T. Scheffer. 2011, pp. 1169–1176.

[219] S. Abuelsamid. "Nvidia Opens up a Lead in Compute System for Automated Driving". *Forbes* (2018). (Accessed on 2019/11/27.)

[220] S. D. Pendleton, H. Andersen, X. Du, et al. "Perception, Planning, Control, and Coordination for Autonomous Vehicles". *Machines* 5.1 (2017).

[221] G. Andrew and J. Gao. "Scalable Training of L1-Regularized Log-Linear Models". *ACM International Conference on Machine Learning (ICML)*. New York, NY, USA, 2007, pp. 33–40.

[222] L. J. Ratliff, S. A. Burden, and S. S. Sastry. "On the Characterization of Local Nash Equilibria in Continuous Games". *IEEE Transactions on Automatic Control* 61.8 (2016), pp. 2301–2307.

[223] W. LLC. "Waymo Open Dataset: An Autonomous Driving Dataset". 2019.

[224] R. Kesten, M. Usman, J. Houston, et al. "Lyft Level 5 AV Dataset 2019". 2019.

[225] J. F. Fisac, N. F. Lugovoy, V. Rubies-Royo, et al. "Bridging Hamilton-Jacobi Safety Analysis and Reinforcement Learning". *IEEE International Conference on Robotics and Automation (ICRA)*. 2019, pp. 8550–8556.

[226] N. Heess, D. TB, S. Sriram, et al. "Emergence of Locomotion Behaviours in Rich Environments" (2017). arXiv: `1707.02286 [cs]`.

[227] S. Levine, C. Finn, T. Darrell, and P. Abbeel. "End-to-End Training of Deep Visuomotor Policies". *Journal of Machine Learning Research* 17.39 (2016), pp. 1–40.

[228] J. Barraquand and J. .-.-C. Latombe. "Nonholonomic Multibody Mobile Robots: Controllability and Motion Planning in the Presence of Obstacles". *Algorithmica* 10.2 (1993).

[229] D. S. Yershov and S. M. LaValle. "Sufficient Conditions for the Existence of Resolution Complete Planning Algorithms". *Workshop on the Algorithmic Foundations of Robotics (WAFR)*. Ed. by D. Hsu, V. Isler, J.-C. Latombe, and M. C. Lin. Springer Tracts in Advanced Robotics. Berlin, Heidelberg: Springer, 2011, pp. 303–320.

[230] R. J. Williams. "Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning". *Machine Learning* 8.3 (1992), pp. 229–256.

[231] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine. "Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor". *ACM International Conference on Machine Learning (ICML)*. 2018, pp. 1861–1870.

[232] A. K. Akametalu, S. Ghosh, J. F. Fisac, and C. J. Tomlin. "A Minimum Discounted Reward Hamilton-Jacobi Formulation for Computing Reachable Sets". *Under review in IEEE Transactions on Automatic Control.* (2018).

[233] R. S. Sutton. "Learning to Predict by the Methods of Temporal Differences". *Machine Learning* 3.1 (1988), pp. 9–44.

[234] J. N. Tsitsiklis. "Asynchronous Stochastic Approximation and Q-Learning". *Machine Learning* 16.3 (1994), pp. 185–202.

[235] G. Brockman, V. Cheung, L. Pettersson, et al. "OpenAI Gym" (2016). arXiv: `1606.01540 [cs]`.

[236] Q. Fischer, C. Hesse, S. Hashme, et al. "OpenAI Five". 2018.

[237] E. Todorov, T. Erez, and Y. Tassa. "MuJoCo: A Physics Engine for Model-Based Control". *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2012, pp. 5026–5033.

[238] V. Rubies-Royo, D. Fridovich-Keil, S. Herbert, and C. J. Tomlin. "A Classification-Based Approach for Approximate Reachability". *International Conference on Robotics and Automation (ICRA)*. 2019, pp. 7697–7704.