Approaches to Safety in Inverse Reinforcement Learning

Dexter Scobee

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2020-8
http://www2.eecs.berkeley.edu/Pubs/TechRpts/2020/EECS-2020-8.html

January 10, 2020

Approaches to Safety in Inverse Reinforcement Learning

by

Dexter Ryan Richard Scobee

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, Chair
Professor Anca D. Dragan
Professor Koushil Sreenath

Fall 2019

Approaches to Safety in Inverse Reinforcement Learning

Abstract

Approaches to Safety in Inverse Reinforcement Learning

by

Dexter Ryan Richard Scobee

Doctor of Philosophy in Engineering - Electrical Engineering and Computer Sciences

University of California, Berkeley

Professor S. Shankar Sastry, Chair

As the capabilities of robotic systems increase, we move closer to the vision of ubiquitous robotic assistance throughout our everyday lives. In transitioning robots and autonomous systems from traditional factory and industrial settings, it is critical that these systems are able to adapt to uncertain environments and the humans who populate them. In order to better understand and predict the behavior of these humans, Inverse Reinforcement Learning (IRL) uses demonstrations to infer the underlying motivations driving human actions. The information gained from IRL can be used to improve a robot's understanding of the environment as well as to allow the robot to better interact with or assist humans.

In this dissertation, we address the challenge of incorporating *safety* into the application of IRL. We first consider safety in the context of using IRL for assisting humans in shared control tasks. Through a user study, we show how incorporating haptic feedback into human assistance can increase humans' sense of control while improving safety in the presence of imperfect learning. Further, we present our method for using IRL to automatically create such haptic feedback policies from task demonstrations.

We further address safety in IRL by incorporating notions of safety directly into the learning process. Currently, most work on IRL focuses on learning explanatory *rewards* that humans are modeled as optimizing. However, pure reward optimization can fail to effectively capture hard requirements, such as safety *constraints*. We draw on the definition of safety from Hamilton-Jacobi reachability analysis to infer human perceptions of safety and to modify robot behavior to respect these learned safety constraints. We also extend this work on learning constraints by adapting the framework of Maximum Entropy IRL in order to learn hard constraints given nominal task rewards, and we show how this technique infers the most likely constraints to align expected behavior with observed demonstrations.

To my family.
Thank you for always encouraging and inspiring me.

In memory of Professor Andrew K. Packard,
who never stopped serving his students.

Contents

Contents **ii**

List of Figures **iv**

List of Tables **v**

I Preliminaries **1**

1 Introduction **2**
1.1 Emerging Human Robot Interactions 2
1.2 Automating Adaptation 3
1.3 Safety and Inverse Reinforcement Learning 4
1.4 Contributions 5

2 Background **7**
2.1 Reachability for Safety 7
2.2 Shared Control 12
2.3 Inverse Reinforcement Learning 17

II Assisting Safely **25**

3 Haptic Assistance via Inverse Reinforcement Learning **26**
3.1 Related Work 27
3.2 Haptic Assistance in IRL-based Shared Control 29

4 Improving Shared Control with Haptic Feedback **35**
4.1 Study on Haptic Assistance Preferences 35
4.2 Analysis and Discussion 38
4.3 Conclusion 41

III Learning Safety **42**

5 Supervisor Safe Sets **43**
5.1 Supervisor Safe Set Control 45
5.2 Experimental Design for User Validation 53
5.3 Analysis and Discussion 56
5.4 Conclusion 59

6 Maximum Likelihood Constraint Inference **61**
6.1 Related Work 62
6.2 Maximum Likelihood Constraint Inference 63
6.3 Examples 73
6.4 Conclusion and Future Work 78

IV Conclusions **79**

7 Future Directions **80**

8 Conclusion **83**

Bibliography **85**

List of Figures

2.1 Relationship between reachability payoff and value functions 9
2.2 Illustration of sets from Reachability Analysis 10
2.3 Representation of Artificial Force Fields 14
2.4 Representation of Virtual Fixtures 15
2.5 Linear policy blending with and without online inference 16
2.6 State machine representation of an MDP 18
2.7 Basic example of IRL on a grid world 20

3.1 Components of providing haptic assistance via IRL 34

4.1 Roadmaps for haptic assistance user study 36
4.2 Objective measures from haptic assistance user study 39
4.3 Subjective measures from haptic assistance user study 40

5.1 Perceived Safety 44
5.2 Reachability with Dubins Car dynamics 46
5.3 Zero level sets for a library of dynamics functions 50
5.4 Example intervention data and value estimation 51
5.5 Safe sets from supervisor user study 53
5.6 Task screen from supervisor user study 54
5.7 Supervisory false positives by safe set 57
5.8 Learned vs. Baseline safe sets 58
5.9 Empirical distribution of supervisor interventions 60

6.1 Illustration of constraints modifying an MDP 65
6.2 Illustration of adding constraints to maximize demonstration likelihood 69
6.3 Maximum Likelihood Constraint Inference on a synthetic grid world MDP . . . 74
6.4 Parametric analysis of the effect of $d_{D_{\mathrm{KL}}}$ on constraint inference performance . . 76
6.5 Maximum Likelihood Constraint Inference on human demonstrations 77

List of Tables

5.1 Predicted vs. observed false positives in supervisor user study 59

Acknowledgments

I would first like to thank my research advisor, Professor Shankar Sastry, for providing me with the freedom to explore my academic interests and the support and guidance needed to follow through on them. I also thank the other members of my dissertation committee: Professor Anca Dragan, who helped catalyze my exploration of algorithmic human-robot interaction, and Professor Koushil Sreenath, who is always a great source of ideas for advancing autonomy. I extend a special thanks to Professor Andrew Packard, one of my first instructors at Berkeley and a shining light who will be missed in the community.

As a whole, I found the Berkeley community to be extremely open and collaborative, encouraging everyone to share ideas and succeed together, and I consider myself lucky to have studied in such a supportive environment. I am grateful to all of my labmates, classmates, officemates, and all other *mates for building this environment, from seminars to prelim prep to barbecues, which really made the Berkeley experience. In particular, the research in this dissertation and beyond would not have been possible without my wonderful collaborators. While a Ph.D. involves much individual effort, I most enjoyed the time spent with others, brainstorming ideas and conquering deadlines. I thank my coauthors, Roy Dong, Lillian Ratliff, Henrik Ohlsson, Michel Verhaegen, Sam Burden, Ryan Robinson, Shankar Sastry,[1] David McPherson,[1] Joe Menke,[1] Allen Yang,[1] Vicenç Rubies Royo,[1] Claire Tomlin,[1] Jaime Fernández Fisac, Anca Dragan, and Andreea Bobu, for making research a team sport.

I would be remiss if I did not also thank the extended team of people who enabled this research. Particularly, Jessica Gamble and Shirley Salanio were instrumental in helping me navigate the requirements of grad school with minimal stress.

Of course, it has also been a privilege to be funded for conducting research that I am so passionate about, and I must thank the Office of Naval Research, who funded this work under the Embedded Humans MURI (N00014-13-1-0341), and the National Science Foundation, who funded this work under the VeHICaL Grant (CNS-1545126).

If research is one side of the academic coin, then teaching is the other. I found teaching to be a fulfilling and rewarding part of my academic journey, and that was in no small part because I worked with excellent instructors, Professors Claire Tomlin and Ruzena Bajcsy, and excellent co-GSIs, Sarah Seko and Oladapo Afolabi.

Beyond Berkeley, I am thankful for all of the other people who have supported me throughout this journey. I have been so fortunate to build and retain strong friendships throughout my life, and I do not think that anyone else has ever been quite so lucky when it comes to roommates. My current roommate (and wife), Ilina Mitra, has been a constant source of love, encouragement, and distraction, all in the proper amounts, and she has played an outsized role in helping me follow my dreams (in a timely manner). I cannot thank her enough. Finally, I am grateful for all of my family, who have supported me not just through this Ph.D., but through all my endeavors over the past 30 years. Thank you all, not only for encouraging me, but for inspiring me too.

[1] Contributed directly to work appearing in this dissertation (i.e., an antecedent of the ubiquitous "we")

Part I

Preliminaries

Chapter 1

Introduction

Advances in mechanical design and artificial intelligence continue to expand the horizons of robotic applications. As robots become more capable, there are increasing opportunities to deploy them outside of traditional, controlled industrial settings. However, as we expand the scope of what robots can do, we have a responsibility to give special consideration to the types of environments in which these robots will operate and the roles that they will play there. If the dream of robotics is to have ubiquitous, automated assistants present all throughout our daily lives, then achieving this dream relies heavily on our ability to design robots that can successfully and safely operate around, and in coordination with, humans. As automation becomes more pervasive throughout society, humans will increasingly find themselves interacting with these autonomous and semi-autonomous systems, and this expectation has driven activity in the field of Human-Robot Interaction (HRI).

In order to facilitate effective HRI, robots must be able to learn from and adapt to humans so that they can improve task performance and carry out those tasks in human-compatible ways. To achieve these goals, techniques in Inverse Reinforcement Learning (IRL) have been developed to try to understand the underlying motivations of human behavior [1]–[5]. However, while these approaches to IRL have achieved impressive results, they do not explicitly recognize the role that *safety* plays in defining desirable behavior. As we allow robotic systems more freedom to interact with humans in the world, such as by driving alongside us, we must ensure that they understand the *hard constraints* that humans obey to ensure safety. In this dissertation, we address the challenge of incorporating safety into the application of IRL in order to create more capable and human-compatible robots in exciting emerging applications.

1.1 Emerging Human Robot Interactions

The successes of automation technology in fields such as aviation, industrial manufacturing, or robot-assisted surgery may make it seem as though the challenges of HRI have been "solved." However, there is an important distinction between these fields and emerging

robotic applications: the level of predictability of the environments and the level of training of the humans with which the robotic systems interact. Robots in industrial settings are traditionally designed to perform a single task many, many times in almost exactly the same way. The environment around the robot is carefully controlled to prevent any unexpected occurrences. In flight, too, the environment is generally predictable due to the relative emptiness of airspace and strict air traffic control regulations.

Moreover, pilots, factory workers, and surgeons are trained professionals who spend their careers learning about and interacting with specific systems. Through practice, these experts are able to adapt to the peculiarities of the systems with which they interact to achieve mastery. In contrast, emerging robotic applications target environments with non-professional, non-expert humans, such as care-giver robots assisting patients in hospitals or at home, or autonomous vehicles managing interactions with passengers, pedestrians, and human-driven vehicles.

In these new applications, the burden of adaptation will shift from humans to the autonomous systems themselves. Users may only have a single interaction with a particular robot, but these robots will have many interactions with different humans, and they can have even more experience if data is shared among networks of robots. These experiences will enable them to learn about human behavior over time. By using this knowledge to adapt their behavior, these systems can increase the facility and fluidity of their interactions with humans, making them a more seamless component of our lives.

1.2 Automating Adaptation

With the liberation of robots from carefully controlled environments comes the need for them to handle uncertainty and multitudes of possible outcomes. The task of "manually" designing and programming a robot to handle all of these possibilities can range from daunting to impossible. By instead building these systems to adapt, engineers can free themselves from the burden of future-proofing these systems at design time.

From a software perspective, this adaptation can be achieved via machine learning, which, broadly, endeavors to make use of the available information (e.g. observations of the environment, human behavior / commands, task outcomes, etc.) to learn how to improve the performance of an automated system. More specifically, IRL uses observations of human behavior to infer human preferences and motivations. Typically, these elements are captured in the form of a reward function that, generally speaking, assigns positive values to behaviors that humans perform and negative values to behaviors that humans avoid. By using this learned reward to govern its own behavior, the robot will become more "human-compatible", either by imitating human behavior to accomplish the same task or by having better predictions of what the human intends to do to facilitate interaction.

One drawback of reward-based IRL is that it is not designed to capture *hard* constraints from the demonstrated behavior. While rewards can capture *soft* constraints, behaviors that are discouraged, they cannot describe behaviors that *must not* occur. In domains

where robots operate in coordination with and in proximity to humans, maintaining safety is an imperative most succinctly described by hard constraints. In this work, we present research that takes a step towards ensuring that safety is accounted for in IRL.

1.3 Safety and Inverse Reinforcement Learning

Robots employed in industrial settings are subject to regulations put forth by entities such as the International Organization for Standardization (ISO) and the American National Standards Institute (ANSI). These regulations put requirements on how robotic systems can be deployed, with an eye to maintaining the safety of human workers. Some of the most commonly applied designs for compliance include placing physical barriers around the robot's workspace and implementing emergency shut-offs whenever a human approaches the workspace [6], [7]. Clearly, these approaches to safety will be infeasible for new applications where the robot's workspace is, by design, populated by humans.

While newer versions of these regulations have been put forward that account for mobile robots and more interactive "cobots," their focus remains on defining safe limits for robot speed and interaction forces for robots in industrial settings [6]–[8]. These are certainly important considerations, but we must also consider how safety is incorporated into the learning elements that will be necessary in new, non-industrial robotic applications. Consider the case of a shared control task, discussed further in Section 2.2, where a human is assisted by an autonomous agent, such as advanced driver assistance capabilities in semi-autonomous vehicles. If the assistance is designed to be adaptive through the application of, for example, IRL, how can we ensure that the learned assistance will lead to safe interactions?

It may not be possible to offer any firm guarantees of safety when humans are integrated into a control loop, but there are approaches that can improve the human's experience of control and reduce the likelihood of unintended errors. Specifically, prior work has shown that *haptic feedback*, assistive forces and torques applied to a human's control interface, can improve the shared control experience and task performance [9]–[11]. The results suggest that it would be advantageous for learning-enabled assistive systems to be able to provide the same type of haptic feedback. To this end, in Chapter 3 we propose a method for turning task demonstrations directly into assistive haptic feedback via IRL.

Further, it is possible for learning-enabled assistive systems to have imperfect learning or to incorrectly infer a human user's intent. Previous work has shown that, under these conditions, humans prefer assistance methods that afford them a greater degree of control over the system [12]. In Chapter 4, we explore this result by examining whether the use of haptic feedback can provide this sense of control and improve performance and safety in the presence of imperfect assistance policies.

While incorporating haptic feedback into IRL-based assistance can help address safety when learning is being applied, it does not modify the common reward-based formulation of current IRL methods. As mentioned above in Section 1.2, learning reward functions is not always enough. Sometimes there are behaviors that should not only have a negative value,

but which simply should *not* occur. One such example is in obstacle avoidance, where a collision is not only undesirable, it is catastrophic. To address this issue, in Chapter 5 we develop a method of learning about the human's notions of safety by querying the human about their perception of a robot performing a task. This approach is based on the definition of safety from Hamilton-Jacobi reachability analysis, which allows us to learn a principled representation of how the human might perceive the safety of the system. We discuss how we can use this learned information to modify a robot's behavior to better align with the human's notion of safety, and we show the technique leading to improved performance on a human-robot teaming task.

One difficulty with this approach, however, is that we only gain information about the human's perceived constraints when things are going poorly. That is to say, in order to obtain an observation where the human views the system to be unsafe, it must be the case that the human is currently viewing the system's behavior as unsafe. To address this point, we develop a method, presented in Chapter 6, that learns these types of constraints from demonstrated behavior. This approach adapts the formulation of Maximum Entropy IRL [5] so that, rather than learning a reward function, we can use a nominal reward to learn hard constraints from safe, successful demonstrations. Using this new paradigm, we are able to infer the most likely constraints motivating the demonstrated behavior.

1.4 Contributions

The techniques mentioned above in Section 1.3 represent our effort in this dissertation to better enable safety considerations to be integrated into IRL-enabled applications. We summarize the contributions of this dissertation as follows:

- We develop a method for automatically converting expert demonstrations into haptic feedback assistance policies via IRL.[1]
- We present a user study demonstrating that providing IRL-based assistance via haptic feedback can improve humans' sense of control and the safety of the system in the presence of imperfect learning.[1]
- We incorporate formal notions of safety from Hamilton-Jacobi reachability analysis into a framework for learning human safety preferences.[2]
- We adapt the approach of Maximum Entropy IRL to perform Maximum Likelihood Constraint Inference, which allows us to infer safety, or other, constraints from demonstrations.[3]

[1] We first presented this work in "Haptic Assistance via Inverse Reinforcement Learning" [13].

[2] We first presented this work in "Modeling Supervisor Safe Sets for Improving Collaboration in Human-Robot Teams" [14].

[3] We first presented this work in "Maximum Likelihood Constraint Inference for Inverse Reinforcement Learning" [15].

By studying how to improve IRL-based assistance with haptic feedback, the first two represent approaches to assisting *safely*, and are detailed in Part II. The second two focus on integrating notions of safety directly into the learning process, and thus constitute approaches to learning *safety*. We present these methods in Part III. Taken together, these contributions represent approaches to integrating safety considerations into IRL-enabled robotic applications and human-robot interactions.

Chapter 2

Background

2.1 Reachability for Safety

In its broadest sense, reachability analysis looks to answer the questions "starting from here, what can be reached?" and "if we want to reach there, where can we start?" In this work, we focus specifically on Hamilton-Jacobi (HJ) reachability, a branch of optimal control that seeks to describe the set of possible outcomes for a dynamical system under bounded disturbances. The HJ approach allows us to answer the two primary questions of reachability analysis by computing the set of states reachable from a given initial condition or the set of states that can reach a given terminal condition.

To compute these sets, HJ formulates the behavior of a dynamical system as the result of a game played between a controller and a disturbance. The controller is designed to keep the system out of an undesirable set of states and/or to reach a set of goal states. While the disturbance typically arises due to stochasticity in the environment or modeling errors in the system, it is usually modeled as being adversarial to the controller for conservatism.

We are primarily concerned with using reachability to evaluate safety, so we focus on controllers avoiding undesirable states. Maintaining safety thus equates to respecting state-space *constraints* by avoiding these prohibited regions. In this setting, HJ reachability can be used to compute *safe sets* of states, from which the model guarantees that there exists a controller that can preserve safety if it begins driving the system from within the safe set. Moreover, computing this safe set also reveals the policy of such a controller, which can then be utilized to enforce safety. We describe the HJ reachability approach to safety in mathematical detail below.

2.1.1 Dynamical Systems

Consider a dynamical system with bounded control input u and bounded disturbance d, given by

$$\begin{aligned} \dot{x} &= f(x,u,d), \\ x \in \mathbb{R}^n, \quad u \in \mathcal{U} \subset \mathbb{R}^{n_u}, \quad d \in \mathcal{D} \subset \mathbb{R}^{n_d}, \end{aligned} \tag{2.1}$$

where $\mathcal{U}$ and $\mathcal{D}$ are compact. We let $\mathscr{U}$ and $\mathscr{D}$ denote the sets of measurable functions $\mathbf{u} : [0,\infty) \to \mathcal{U}$ and $\mathbf{d} : [0,\infty) \to \mathcal{D}$, respectively, which represent possible time histories for the system input and disturbance. Given a choice of input and disturbance signals, there exists a unique continuous trajectory $\zeta : [0,\infty) \to \mathbb{R}^n$ from any initial state x which solves

$$\begin{aligned} \dot{\zeta}(t) &= f(\zeta(t), \mathbf{u}(t), \mathbf{d}(t)), \text{ a.e. } t \geq 0, \\ \zeta(0) &= x, \end{aligned} \tag{2.2}$$

where $\zeta(\cdot)$ describes the evolution of the dynamical system [16]. For clarity, we let $\zeta_{x,t_0}^{\mathbf{u},\mathbf{d}}(\cdot)$ specifically denote the trajectory of the system beginning in initial state x at time t_0, being driven by input $\mathbf{u}(\cdot)$ and disturbance $\mathbf{d}(\cdot)$.

2.1.2 Safe Sets

Obstacles in the environment can be modeled as a "keep-out" set of states $\mathcal{K} \subset \mathbb{R}^n$ that the system must avoid. We define the safety of the system with respect to this set, such that the system is considered to be safe at state $\zeta(0) = x$ over time horizon T as long as we can choose $\mathbf{u}(\cdot)$ to guarantee that there exists no time $t \in [0,T]$ for which $\zeta(t) \in \mathcal{K}$. The task of maintaining the system's safety over this interval can be modeled as a differential game between the control input and the disturbance.

When formulating this game, we want to ensure that we produce conservative estimates of what the control can achieve, such that the outcome of the game provides valid guarantees on safety. To this end, we follow the convention of modeling the disturbance as having *instantaneous* knowledge of the control input but restricting it to play *non-anticipatively* [17]–[19]. In this framework, the disturbance can be written as a function of the control input, such that $\mathbf{d}(\cdot) = \beta[\mathbf{u}](\cdot)$, where $\beta : \mathscr{U} \to \mathscr{D}$ is restricted to the set of non-anticipative strategies $\mathcal{B}$, given by [19]:

$$\begin{aligned} \mathcal{B} = \{\beta : \mathscr{U} \to \mathscr{D} \mid \forall t \geq 0,\ \forall \mathbf{u}(\cdot), \hat{\mathbf{u}}(\cdot) \in \mathscr{U}, \\ \mathbf{u}(\tau) = \hat{\mathbf{u}}(\tau) \text{ a.e. } \tau \geq 0 \implies \beta[\mathbf{u}](\tau) = \beta[\hat{\mathbf{u}}](\tau) \text{ a.e. } \tau \geq 0\}. \end{aligned} \tag{2.3}$$

By requiring that $\beta \in \mathcal{B}$, the disturbance must not respond differently to any control signals until the signals themselves diverge, effectively preventing the disturbance from *anticipating* future control inputs.

We further define the objective of this game by choosing any Lipschitz payoff function $l : \mathbb{R}^n \to \mathbb{R}$ which is negative-valued for $x \in \mathcal{K}$ and positive for $x \notin \mathcal{K}$. The control input

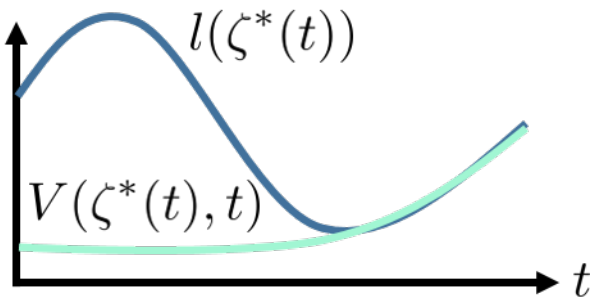


Figure 2.1: Relationship between reachability payoff function l and value function V. This cartoon shows their relationship as described by (2.4): the value of $V(\zeta^*(t),t)$ at any point is dictated by the lowest value of $l(\zeta^*(t))$ found in the future. When the system is currently at the lowest payoff that will be encountered, the payoff and value functions are equal.

attempts to maximize this payoff, avoid $\mathcal{K}$, while the disturbance attempts to minimize this payoff, enter $\mathcal{K}$. We define the *outcome* of this game as the minimum value of $l(x)$ achieved over the time horizon $[0,T]$, such that a positive-valued outcome indicates successful avoidance of $\mathcal{K}$ and a negative-valued outcome indicates a collision with $\mathcal{K}$. The *value* of this game at any state x and point in time $t \in [0,T]$ is the outcome achieved from that point by an optimal input and disturbance, given by:

$$V(x,t) = \inf_{\beta \in \mathcal{B}} \sup_{\mathbf{u}(\cdot) \in \mathscr{U}} \min_{\tau \in [t,T]} l(\zeta_{x,t}^{\mathbf{u},\beta[\mathbf{u}]}(\tau)). \tag{2.4}$$

Solving (2.4) will produce an optimal control signal $\mathbf{u}(\cdot)$ which attempts to steer the system away from $\mathcal{K}$ and an optimal disturbance $\mathbf{d}(\cdot)$ which attempts to drive the system towards $\mathcal{K}$. The relationship between the value function and the payoff function described here is illustrated in Figure 2.1.

We can further characterize this value function by the following Hamilton-Jacobi-Isaacs variational inequality [17]:

$$\begin{aligned} &\min \begin{cases} l(x) - V(x,t), \\ \frac{\partial V}{\partial t}(x,t) + \max_{u \in \mathcal{U}} \min_{d \in \mathcal{D}} \frac{\partial V}{\partial x}(x,t) \cdot f(x,u,d) \end{cases} = 0, \\ &V(x,T) = l(x). \end{aligned} \tag{2.5}$$

The value function $V(x,t)$ that satisfies the above conditions is the solution to (2.4) and is equal to $\min_{\tau \in [t,T]} l(\zeta^*(\tau))$ for the trajectory with $\zeta^*(t) = x$ driven by an optimal control $\mathbf{u}(\cdot)$ and an optimal disturbance $\mathbf{d}(\cdot)$. We can therefore find the set of states $\mathcal{R}_T = \{x \in \mathbb{R}^n : V(x,0) < 0\}$ from which we cannot guarantee the safety of the system on the interval $[0,T]$, also known as the backward reachable tube of $\mathcal{K}$ over this interval. That is, for all initial states $x \in \mathcal{R}_T$ and feedback control polices $\mathbf{u}(t) = g(\zeta(t))$, there exists some disturbance $\mathbf{d}(\cdot) = \beta[\mathbf{u}](\cdot)$, with $\beta \in \mathcal{B}$, such that $\zeta(t) \in \mathcal{K}$ for some $t \in [0,T]$.

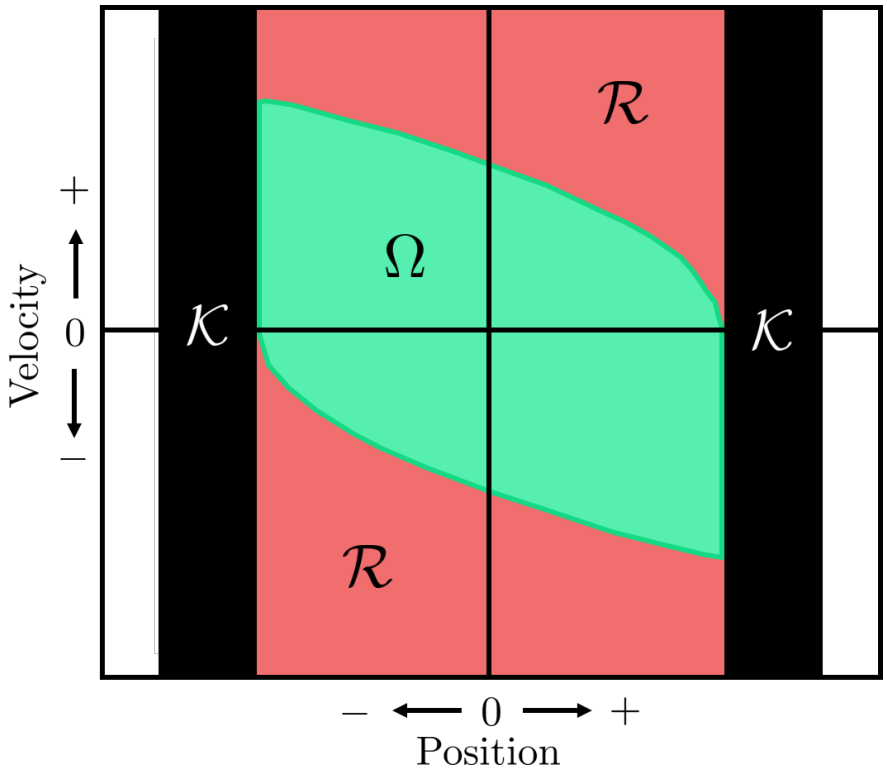


Figure 2.2: Illustration of the relationship between a keep-out set $\mathcal{K}$, the derived backward reachable tube $\mathcal{R}$, and the resulting safe set Ω. Note that $\mathcal{K} \subseteq \mathcal{R}$, and Ω is equal to the complement of $\mathcal{R}$. This illustration approximates the result obtained using a simple double integrator system with $\ddot{x} = u$, $u \in [-u_{max}, u_{max}]$, where $\mathcal{K}$ is given by two walls that bound the space to the left and right. Here, $\mathcal{R}$ is the set of states from which the acceleration u_{max} is insufficient to prevent a collision with a wall in $\mathcal{K}$. Notice that as the system approaches a wall, it must slow down to maintain safety.

If there exists a non-empty controlled-invariant set Ω that does not intersect $\mathcal{K}$, then we deem this set Ω a "safe set" because there exists a feedback policy that guarantees that the system remains in Ω, and thus out of $\mathcal{K}$, for all time. It follows from their properties that Ω is the complement of $\mathcal{R}_T$, and the relationship between $\mathcal{K}$, $\mathcal{R}_T$, and Ω is visualized in Figure 2.2. Within a safe set Ω, the value function becomes independent of t as $T \to \infty$ [17]. Because we focus on the case where the system is initialized to some safe state $\zeta(0) \in \Omega$ and we aim to maintain $\zeta(t) \in \Omega$ for all $t \in [0, \infty)$, we simplify notation by defining the terms $V(x) \triangleq \lim_{T\to\infty} V(x, \cdot)$ and $\mathcal{R} \triangleq \mathcal{R}_\infty$. The final expression for the safe set Ω is given by

$$\Omega = \{x \in \mathbb{R}^n : V(x) \geq 0\}. \tag{2.6}$$

2.1.3 Minimally Invasive Safe Control

One approach to guaranteeing the safety of the system is to apply a "minimally invasive" safety controller which activates on the zero level set of $V(x)$ [20]. This approach allows complete flexibility of control as long as $\zeta(t) \in \text{interior}(\Omega)$, and applies the optimal control to avoid $\mathcal{K}$ when $\zeta(\cdot)$ reaches the boundary of Ω.

Fortunately, computing the optimal avoidance control is tied closely to computing the value function. Considering (2.5), we see that if $l(x) > V(x,t)$, implying that the system will approach $\mathcal{K}$ more closely in the future, then it must be the case that the total change in the value function must be equal to 0 under the optimal control and disturbance. We note that this observation makes intuitive sense: the value function captures the outcome of the game under optimal control and disturbance, so if the system is driven optimally, the value should remain constant unless the system is already at the point of closest approach. Therefore, the optimal control input u_x^* at state x is given by the solution to the optimization in (2.5). Considering that we are interested in the case where the safe set converges and becomes independent of time, we can express this input as:

$$u_x^* = \underset{u \in \mathcal{U}}{\operatorname{argmax}} \min_{d \in \mathcal{D}} \frac{\partial V}{\partial x}(x) \cdot f(x,u,d). \tag{2.7}$$

In other words, this optimal input represents the input value that drives the system dynamics towards the best worst-case change in the value function, which will always be greater than or equal to zero. Therefore, if we apply this control at the boundary of Ω, when $V(x) = 0$, then the system will remain in Ω since this input will prevent the value function from decreasing below zero.

If we applied this optimal input everywhere, that is $\mathbf{u}(t) = g^*(\zeta(t)) = u^*_{\zeta(t)}$, we would achieve the optimal avoidance policy. In general, we will have other goals for the system besides just obstacle avoidance, so it is impractical to simply select the policy $g^*(\cdot)$ for control. Let us denote the policy designed to achieve these other objectives by $g_0(\cdot)$. We can then define the minimally invasive control policy $g_{\text{mi}}(\cdot)$ as

$$g_{\text{mi}}(x) = \begin{cases} g_0(x) & \text{if } V(x) > 0 \\ g^*(x) & \text{otherwise} \end{cases}. \tag{2.8}$$

As desired, this minimally invasive control allows flexibility while the system remains within the safe set but guarantees safety by applying the optimal avoidance controller just before the system exists the safe set. We can even consider $g_0(\cdot)$ to be provided by a human operator, in which case following $g_{\text{mi}}(\cdot)$ allows the human to maneuver the system however they see fit as long as they do not attempt any unsafe behavior, as described in [20]. We refer the interested reader to [17], [18], [20] for a more thorough treatment of reachability and minimally invasive controllers.

2.2 Shared Control

When controlling or teleoperating a robotic system, a human user will provide input commands designed to achieve some goal behavior. While directly implementing the user's input signal may provide the greatest degree of control over the system, the user's control signal may not correctly reflect the user's *intent*. Such errors in control could be the result of high-level phenomena, such as the user incorrectly perceiving or interpreting the system state [21], low-level phenomena, such as deficiencies in the input mechanism or simple noise in the human's movements [22], or a confluence of both [23]. In light of these complications, the performance of the human-robot system can be improved by augmenting the user's input rather than passing it directly to the controlled system.

In shared control, or assistive teleoperation, both a human and an autonomous agent are able to simultaneously influence the behavior of the controlled system [12], [24]–[26]. By allowing the automation to augment the user's input signal, it is possible for the robotic system to better reflect the user's intent, even if the system does not strictly obey the human's commands. Consider the minimally invasive safety control policy discussed in 2.1.3: if we assume that the user never intends to crash (enter the keep-out set $\mathcal{K}$), then this safety controller respects the users intent by ignoring their inputs and applying the optimal safe control policy just before $\mathcal{K}$ becomes unavoidable.

Abbink and Mulder divide such shared control schemes into two broad categories: "input-mixing shared control" and "haptic shared control" [27]. With input-mixing, the automation is able to modify a user's input signal before passing it along to the system's actuators. This type of intervention can usually be modeled as some variation of a policy-blending approach, where the actual system input is constructed based on the user's input and some policy determined by the automation [12], [26]. With haptic shared control, the automation is able to exert forces on the control interface used by the human, thereby influencing the input signal through force interactions with the user [24]. In contrast to input-mixing approaches, haptic approaches maintain the coupling between the configuration of the user's control interface and the behavior of the actuators. In Chapter 4, we examine how employing haptic feedback can improve a user's sense of control and make the safety of the overall system more robust to errors in intent inference.

2.2.1 Intent Inference

A key element of providing assistance to a human, and interacting with humans in general, is to understand what the human *intends* to do. Inferring intent allows the automation to provide useful, rather than counter-productive, assistance. The simplest way to infer intent is to assume that it is constant, that any human interacting with the system will be attempting to accomplish the same task in the same way [28]. A more flexible approach would be to assume that certain *sub-tasks* remain constant while the overall task may change [20], [29]. For instance, the minimally invasive safety controller framework discussed in Section 2.1.3

and deployed in [20] assumes that the human intends to avoid collisions but does not assume any other aspect of the human's intent.

Of course, the most flexible way to provide assistance is to dynamically infer the human's intent as more observations of their behavior become available [20], [29]–[34]. While we could ideally infer any possible goal G, for tractability the inference is typically restricted to a finite hypothesis space $\mathcal{G}$. This space $\mathcal{G}$ could be a finite set of possible goals, or it could consist of goals parameterized by finite-dimensional parameters θ. For example, as is common in IRL, we could describe the human's goal as maximizing a reward function R_θ parameterized by θ. A common example is to let R_θ be a linear combination of features ϕ observed in the human's behavior, such that $R_\theta = \theta^T \phi$, as discussed in Section 2.3.2. Inferring the goal in this case is equivalent to finding a value for the parameters of the reward which can explain the observed behavior.

The problem of inferring a human's goal can in general be restated as finding the most likely goal G^* given the available observations. If these observations reveal the human's behavior as a trajectory $\xi_{[t_0,t]}$ from initial time t_0 to the current time t, then we can find G^* by solving

$$G^* = \underset{G \in \mathcal{G}}{\operatorname{argmax}} P(G \mid \xi_{[t_0,t]}). \tag{2.9}$$

If we further wish to predict *how* the human intends to accomplish this goal, we can formulate the problem as finding the most likely trajectory $\xi^*_{[t,T]}$ given the observations and the inferred goal, as in

$$\xi^*_{[t,T]} = \underset{\xi \in \Xi}{\operatorname{argmax}} P(\xi \mid G, \xi_{[t_0,t]}). \tag{2.10}$$

The problem still remains as to how these probabilities should be defined, or put another way, what is an appropriate probabilistic model of human behavior? This issue is also important to formulating IRL, and we discuss the topic in Section 2.3. To see different approaches to intent inference in more detail, we also refer the interested reader to [12], [30].

2.2.2 Input-Mixing Assistance

Once the human's objectives are understood, assistance can be generated to help them in that endeavor. Let us consider the case where the system being controlled can be modeled as a dynamical system as in 2.1.1. In *input-mixing* shared control, the input applied to this system, u is a function of the input provided by the human, u_H, and the input provided by the automated assistance u_A. If we let g denote the mixing function, then we can express this relationship as

$$u = g(u_H, u_A). \tag{2.11}$$

In a *switched-control* system, the human and the automation may take turns commanding the system depending on the specific sub-task being carried out at a given moment [20], [35], [36]. In this case, the mixing function will switch between $g(u_H, u_A) = u_H$ and $g(u_H, u_A) = u_A$ as necessary depending on which agent should command the current sub-task. The minimally invasive safety controller described by (2.8) follows this paradigm: the

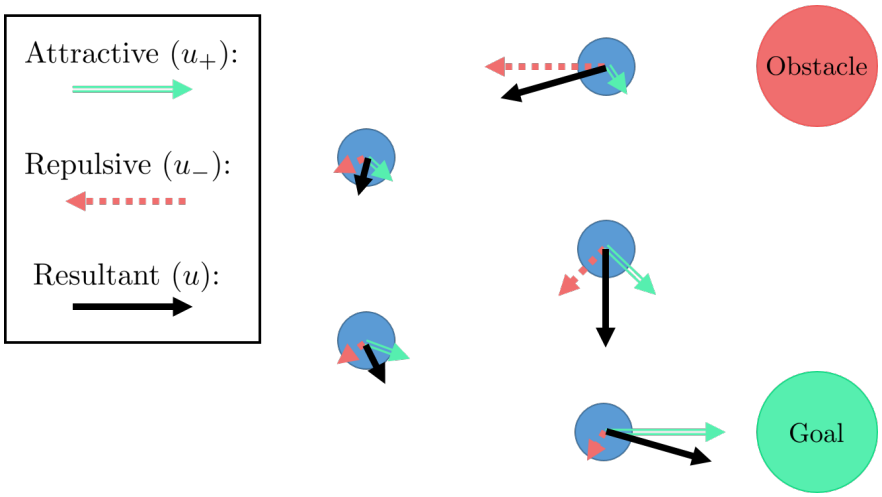


Figure 2.3: Illustration of Artificial Force Fields. The obstacle is modeled as emitting a repulsive force, while the goal emits an attractive force. A robotic system is shown at various locations to illustrate the different resultant inputs u that will be generated by combining the state-dependent repulsive and attractive inputs, u_- and u_+, in the absence of human input u_H.

automated assistance is fully in control when obstacle avoidance is necessary, but the human remains fully in control at all other times.

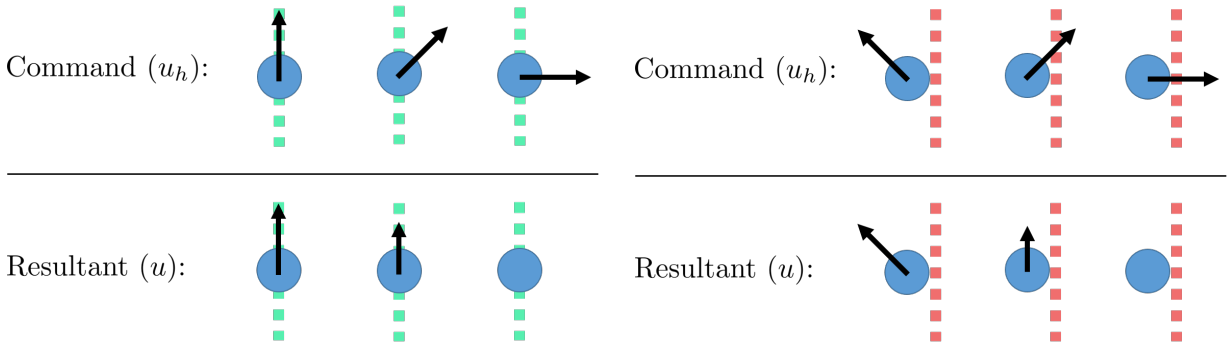
The *artificial force field*, or *potential field*, assistance paradigm can be used to assist a user in avoiding undesirable regions while reaching goal regions [37]–[39]. This method is so called because it is based on treating obstacles as emitting *repulsive* fields and goals as emitting *attractive* fields, as visualized in Figure 2.3. If we consider the net effect on the input due to repulsion as u_- and the effect due to attraction as u_+, then the assistive input u_A will be equal to $u_- + u_+$. The resulting input is generally given by $g(u_H, u_A) = u_H + u_- + u_+$.

Another common paradigm for assistance is known as *virtual fixtures* [28], [40], [41]. Virtual fixtures act as virtual objects in the space which constrain the motion of the system. For example, to prevent motion into a restricted region, a virtual fixture could act as a wall which prevents inputs that could drive the system into that region, as in Figure 2.4b. Alternatively, if we know that the human intends to follow a specific path, then a virtual fixture could act as a guide rail, where only the human input commands along the direction of the fixture are accepted, as in Figure 2.4a.

While there are a variety of approaches to input-mixing assistance, Dragan and Srinivasa present a unifying framework of *linear policy blending* [12]. In this case, the resultant input signal $\mathbf{u} : [0, \infty) \to \mathcal{U}$ is simply a linear combination of the input signal $\mathbf{u}_H$ from the human and the input signal $\mathbf{u}_A$ from the assistance, and can be written as

$$\mathbf{u}(t) = (1 - \alpha(t))\mathbf{u}_H(t) + \alpha(t)\mathbf{u}_A(t), \tag{2.12}$$

where $\alpha : [0, \infty) \to [0, 1]$ is a time-varying *arbitration* function that trades off between



(a) Guidance Virtual Fixture (b) Avoidance Virtual Fixture

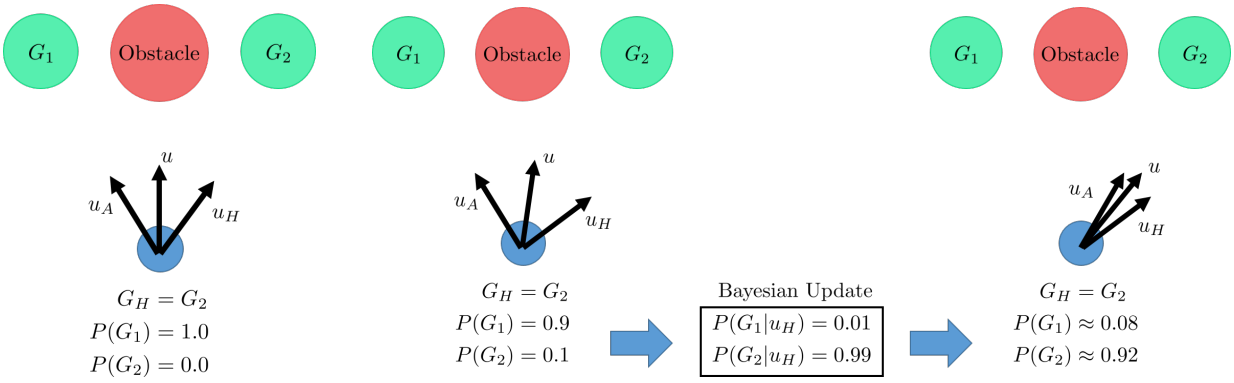
Figure 2.4: Illustration of Virtual Fixtures. In (a), the virtual fixture is employed to restrict motion so that it must lie along a desired path, much like a guide rail. In (b), the virtual fixture is used to prevent motion that would collide with an obstacle, without otherwise restricting motion.

the human's and the assistance's inputs. For example, a switched control assistance policy can be achieved by varying the value of α from the set $\{0, 1\}$ as appropriate depending on the current sub-task. It is the responsibility of the automation to choose an appropriate value of α to best assist the user. As another example, the value of α could vary with the automation's *confidence* in its inference of human intent [12].

When providing assistance, it is important that the assistance is actually useful and not harmful. In the case of using policy blended inputs without on-line goal inference for input-mixing assistance, it is possible for the performance of the assisted system to be worse than either the human or automation acting alone [42]. Figure 2.5a shows an example where the human and the automation disagree on the best course of action, and the resulting blended input leads to a collision. However, as shown in Figure 2.5b active intent inference will allow the automation to update the policy behind u_A, which can help avoid these issues. Applying a haptic-feedback-based assistance scheme can also help alleviate this difficulty by giving the human the ability to directly override the assistance, and we explore how this affects safety and user experiences in Chapter 4.

2.2.3 Haptic Feedback

Haptic feedback spans the gamut from simple vibration alerts to complex force interactions. For our purposes, we are specifically interested in kinesthetic feedback, that is force or torque feedback related to motion, which has been applied in a variety of settings [9]–[11], [43], [44]. Consider, for example, a human and a robot collaborating to lift a heavy load. Coordination between the two agents is key to their success. One natural way for the robot to communicate its intent to the human (and vice-versa) is through the application of force to the shared load. In this scenario, there is a physical linkage between the human and the robot (the



(a) Static Goals (b) Online Goal Inference

Figure 2.5: Illustration of the potential pitfalls of linear policy blending without online intent inference. The figures represent the resultant forces of a simple arbitration function where $u = 0.5u_H + 0.5u_A$. In both figures, the human's true intended goal G_H is to pass to the right of the obstacle, that is $G_H = G_2$. However, the assistance begins with a prior probability that makes passing to the left, G_1, seem more likely. In (a), the assistance does not update its beliefs over possible goals, and the resultant input u drives the system towards the obstacle. In (b), the assistance performs a Bayesian update to its beliefs over possible goals, and the final resultant input avoids the obstacle while driving the system towards correctly towards G_2.

joint load), which provides a natural, intuitive communication channel. This type of haptic communication has several benefits:

- It takes advantage of actuation and sensing already being utilized by the agents.
  - Both the human and the robot are already generating and sensing forces, so other communication channels can remain free (such as vision).
- The messages travel nearly instantaneously between the two agents.
  - Contrast this with speech, which requires additional time both to deliver and process.
- Haptic feedback can be used by the human's fast low-level neuromuscular control, bypassing slower high-level cognition [27].
  - This mode takes advantage of the same neural processes that allow humans to balance and perform athletically.

While there are certainly limitations to the types of information that can be delivered through haptics, this communication channel is naturally suited to processing information relevant to the physical control of dynamical systems.

In the context of teleoperation, the physical linkage between the human and the system being controlled is lost, but haptic feedback between the automation and the human can still be maintained through the use of haptic interface devices, such as the Phantom Premium [45]. Using such haptic feedback for shared control leads to natural implementations of both artificial force fields [11], [29], where a user feels force guiding them toward desirable inputs, and virtual fixtures [40], [43], [44], where a user feels the motion of their input device restricted.

2.3 Inverse Reinforcement Learning

A formulation of the Inverse Reinforcement Learning (IRL) problem was first proposed by Kalman [46] as the Inverse problem of Optimal Control (IOC). Given a dynamical system and a control law, the author sought to identify which function(s) the control law was designed to optimize. This problem was brought into the domain of Markov Decision Processes (MDPs) and Reinforcement Learning (RL) by Ng and Russell [1], who proposed IRL as the task of, given an MDP and a policy (or trajectories sampled according to that policy), find a reward function with respect to which that policy is optimal. The ability to automatically learn a reward function is enticing: while humans may be good at implicitly understanding task goals or even carrying out the tasks ourselves, we can struggle to design a reward function that properly describes a task for a robot [47], [48]. We first define MDPs in the following section, then further discuss approaches to IRL.

2.3.1 Markov Decision Processes

Markov Decision Processes (MDPs) model the interactions of an agent with their environment. The environment includes a set of states S, and the agent is able take actions from a set A. In general, the set of possible actions may vary depending on the state, so we can denote a separate action set for each state s as $A_s \subseteq A$. Based on the current state s of the system, taking an action a will cause a transition to a new state s'. This transition is in general stochastic, and taking action a in state s results in a probability distribution over next states $P_{s,a} : S \to [0, 1]$ such that $s' \sim P_{s,a}$. The initial state of the system is also stochastic in general, so there exists a distribution D_0 over these possible starting states. As the state evolves, the agent observes features in the environment from a feature space Φ. Typically, these features are represented by real-valued vectors that describe the degree to which a feature is present in the environment. The features observed at any time depend on the state and action taken, so we can define a feature map $\phi : S \times A \to \mathbb{R}^{n_\phi}$, where n_ϕ is the number of features in the environment. The agent's behavior, the actions it chooses to take from each state, is motivated by a reward $R : S \times A \to \mathbb{R}$, and that reward is often modeled as depending on the features that the agent observes.

We can combine all of the elements described above to define an MDP $\mathcal{M}$ as a tuple $(S, \{A_s\}, \{P_{s,a}\}, D_0, \phi, R)$. Figure 2.6 shows a simple graphical representation of the states

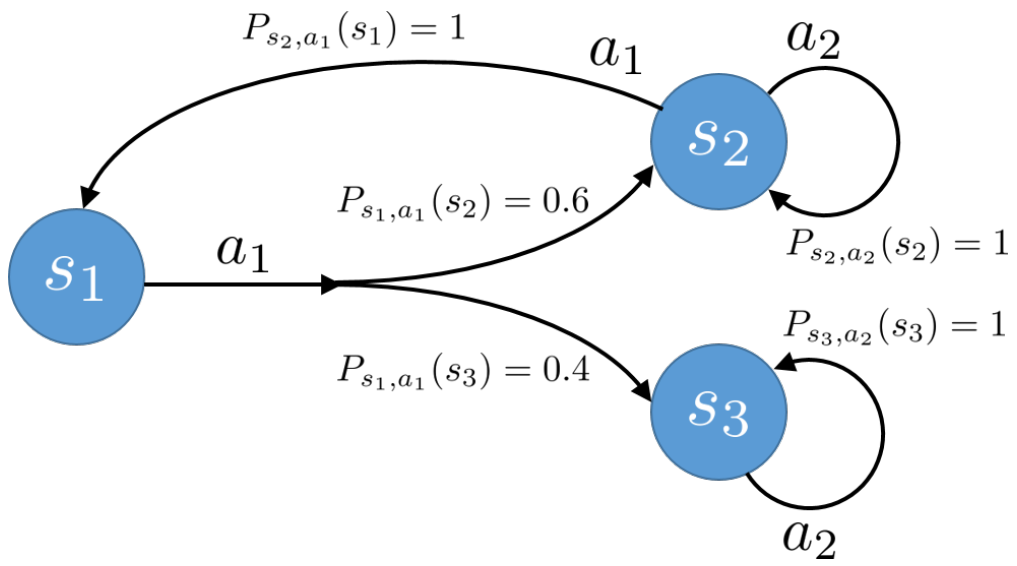


Figure 2.6: A simple state machine representation of a Markov decision process. The circular nodes represent the state space $S = \{s_1, s_2, s_3\}$. The available actions at each state are indicated by arrows, and we can see that $A_{s_1} = \{a_1\}$, $A_{s_2} = \{a_1, a_2\}$, and $A_{s_3} = \{a_2\}$. The transition probabilities $P_{s,a}$ are shown for every state-action pair. The split arrow leaving s_1 indicates a stochastic transition, in accordance with the transition probabilities.

and actions of an MDP. The actions that an agent in an MDP is likely to take in a given state are defined by the agent's policy π, which produces a probability distribution $\pi(\cdot|s, t)$ over actions given the current state s and time t. For time-invariant policies, it is common to drop the explicit dependence on time t. If the policy is deterministic, then it assigns probability one to a single action and probability zero to all others. In this case, the policy can alternatively be expressed as a mapping from states to actions $\pi : S \to A$. By following a policy, the agent will produce a trajectory ξ through the MDP, alternatively referred to as a *trace* ξ. An MDP trajectory ξ consists of one state s_t and one action a_t for every time step in the trajectory, up to the length of the trajectory T, and we can express this as $\xi = \{\mathbf{s}_{0:T}, \mathbf{a}_{0:T}\}$.

As we mentioned earlier, this behavior for the agent is motivated by the reward function R. More specifically, the agent is typically modeled as attempting to maximize the *expected sum of discounted rewards*, which, in the case of an infinite planning horizon,[1] can be expressed as

$$V_\pi(s) = \mathop{\mathbb{E}}_{\xi\sim\pi}\left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t)\right], \tag{2.13}$$

[1]For finite planning horizons, the value functions and policy in (2.13), (2.14), and (2.15) are largely unchanged, though the values are, in general, dependent on the current time, where the summation in (2.13) is modified to run from the current time to the end of the finite planning horizon.

where the expectation is taken over the distribution of trajectories generated by following policy π starting from state s, $\gamma \in [0, 1)$ is a discount factor[2] that de-weights future rewards, and $V_\pi(s)$ is the value function that returns the expected sum of discounted rewards to be earned by an agent following policy π and starting in state s.

The value function V_π is sometimes referred to as the *state* value function, to contrast it with the *action* value function Q_π, given by

$$Q_\pi(s, a) = R(s, a) + \gamma \mathop{\mathbb{E}}_{s' \sim P_{s,a}} \left[V_\pi(s') \right], \tag{2.14}$$

which is the expected value that will be earned by taking action a from state s, then following policy π thereafter. We say that an agent has "solved" an MDP once they have discovered an optimal policy π^* that maximizes the value function everywhere. That is, $\pi^* = \text{argmax}_\pi V_\pi(s)$ for all states $s \in S$. The optimal actions under this policy are given as

$$\pi^*(s) = \mathop{\text{argmax}}_{a \in A_s} Q_{\pi^*}(s, a). \tag{2.15}$$

2.3.2 IRL Formulations

While Reinforcement Learning (RL) uses observations about rewards earned in an environment to find an optimal policy for performing a task, IRL learns about the specifications of a task by using an expert's policy (or more typically a set of demonstrations generated by an expert following that policy) to infer a reward function with respect to which that policy is optimal. In this context, optimality is typically defined with respect to maximizing the value function, as defined in (2.13). In broad terms, IRL can be characterized as solving

$$R^* = \mathop{\text{argmax}}_{R} \mathop{\mathbb{E}}_{s_0 \sim D_0} \left[V_\pi(s_0, 0) \right], \tag{2.16}$$

where the expectations in (2.13) and (2.16) can be replaced with empirical estimates from demonstrations.

One of the chief difficulties in the problem of IRL is the fact that a policy can be optimal with respect to a potentially infinite set of reward functions. The most trivial example of this is the fact that all policies are optimal with respect to a null reward function that always returns zero (or any other constant). Much of the work in IRL has been devoted to developing approaches that address this ambiguity by imposing additional structure on the problem to make it well-posed. In [3], the authors propose considering the class of rewards that are linear in the features of the MDP. Rewards of this form can be parameterized by weights $\theta \in \mathbb{R}^{n_\phi}$, and the reward function can be expressed as

$$R_\theta(s, a) = \theta^T \phi(s, a). \tag{2.17}$$

[2]In finite-horizon planning, the discount factor γ may, and often does, take the value 1 so that the state value corresponds to the expectation of the un-discounted sum of rewards over the horizon.

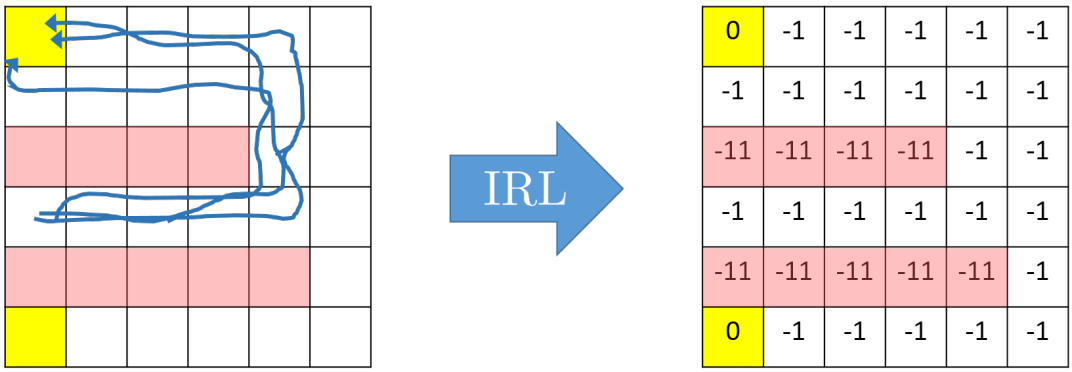


Figure 2.7: A basic illustration of IRL on a grid world MDP. The grid cells represent states, and each state is colored to correspond to the feature earned from taking any action in that state. First, on the left, we do not know the reward function of this MDP, but we have demonstrations of behavior which avoid red states and reach yellow states while passing through as few white states as possible. On the right, after performing IRL, we have learned a feature-based reward function that explains the demonstrated behavior.

With this formulation, solving (2.16) amounts to finding an optimal parameter vector θ^*. Moreover, if we overload the map ϕ to give the feature count of a trajectory as

$$\phi(\xi) = \sum_{t:s_t,a_t \in \xi} \phi(s_t, a_t), \tag{2.18}$$

then we can overload the reward function R_θ to express the cumulative reward earned by a trajectory as

$$R_\theta(\xi) = \theta^T \phi(\xi) \tag{2.19}$$

Under this reward linearity assumption, matching expert performance on an unknown reward function is achieved by finding *any* reward that causes the agent to match the expert's expected feature counts. Figure 2.7 shows an example of a feature-based reward that IRL might learn from a given set of demonstrations.

Later, Ratliff, Bagnell, and Zinkevich [2] presented Maximum Margin Planning (MMP), which cast the problem as one of structured maximum margin prediction, adding a loss term for learned behaviors that deviate from demonstrations, even if they match feature expectations. The MMP approach solves a convex relaxation of this problem to select an approximately optimal reward function to explain demonstrations. By penalizing reward functions that induce the learning agent to match feature expectations but to deviate from the specific demonstrated trajectories, MMP attempts to more closely align the learned reward with the unknown true reward.

Nonetheless, these earlier approaches can struggle when demonstrated behavior is sub-optimal, variable, or inconsistent, as matching feature expectations may require a mixture of optimal policies, and many mixtures may satisfy feature expectations while giving zero probability of reproducing demonstrated behavior. To address these shortcomings, Ziebart, Maas,

Bagnell, *et al.* [5] employ the principle of maximum entropy [49] which allows the authors to develop an IRL algorithm that produces a single stochastic policy that matches feature counts without adding any additional constraints to the produced behavior. This so-called Maximum Entropy IRL (MaxEnt) provides a framework for reasoning about demonstrations from experts who are noisily optimal, with trajectories that produce higher rewards being exponentially more likely, as described in (2.23).

Many of the more recent approaches to IRL leverage this same noisy behavior model, and its induced probability distribution over trajectories, to characterize variations among demonstrations [50]–[52]. In order to apply IRL to problems with large, continuous domains, the Relative Entropy IRL [50] and Guided Cost Learning [52] techniques use sample-based methods to estimate this distribution. The authors of [51] similarly extend the applicability of IRL by using a local approximation of this distribution, which allows them to learn from demonstrations that only need to be *locally* noisily optimal. This maximum entropy probability distribution over trajectories also forms the basis for our efforts in identifying the most likely behavior-modifying constraints in Chapter 6. Due to its pervasiveness in the field and relevance to our work, we give extra attention to the details of the MaxEnt IRL method in the following section.

2.3.3 Maximum Entropy Inverse Reinforcement Learning

Like earlier approaches to IRL, Maximum Entropy IRL (MaxEnt) [5] focuses on selecting a reward function such that behavior optimizing this reward function will have the same expected feature counts as the empirical means observed from demonstrations. In this method, the formulation is based on considering the probability distribution over all possible trajectories in an MDP with discrete state and action spaces. The MaxEnt approach to IRL resolves the ill-posed problem of finding an optimal reward function by invoking the principle of maximum entropy, which dictates that the probability distribution that matches feature expectations, without imposing additional structure, is the distribution that maximizes entropy subject to this matching constraint [49].

Given a set $\mathcal{D}$ that contains N demonstrations ξ, let us denote the expected feature counts estimated from these demonstration as

$$\phi_{\mathcal{D}} = \frac{1}{N} \sum_{\xi \in \mathcal{D}} \phi(\xi). \tag{2.20}$$

Considering a finite set[3] of possible trajectories Ξ, and letting θ parameterize the reward function, the expected feature count induced by a reward function is given as

$$\hat{\phi}(\theta) = \sum_{\xi \in \Xi} P(\xi|\theta)\phi(\xi). \tag{2.21}$$

[3] The set of possible trajectories will be finite when MDPs with finite, discrete state and action spaces are paired with agents acting over a finite horizon. With infinite horizons, the set of possible trajectories can be infinite. However, these infinite sets, as well as large finite sets, can be approximated by a finite subset of trajectories that are "reasonable" according to some domain-specific criteria, as discussed in [5], [53].

As discussed in [49], the probability distribution that maximizes entropy, while ensuring that $\hat{\phi}(\theta) = \phi_{\mathcal{D}}$, will be of the form

$$P(\xi|\theta) \propto e^{\theta^T \phi(\xi)}, \tag{2.22}$$

where the equality of $\hat{\phi}(\theta)$ and $\phi_{\mathcal{D}}$ is achieved by finding the proper value for θ.

Note that the exponent in (2.22) is of the same form as the trajectory reward $R_\theta(\xi)$ given by (2.19). Therefore, the probability distribution in (2.22) will be induced by an agent's policy that makes the likelihood of producing a trajectory exponentially proportional to its cumulative reward:[4]

$$P(\xi|\theta) = \frac{1}{Z(\theta)} e^{R_\theta(\xi)}, \tag{2.23}$$

where $Z(\theta) = \sum_{\xi \in \Xi} e^{R_\theta(\xi)}$ is the *partition function* that normalizes (2.23) to be a proper probability distribution. A stochastic policy that induce this probability distribution can be achieved by utilizing a softened version of (2.15). The maximum entropy policy π_θ for a given θ can be described using the soft value functions[5] presented in [54] and defined recursively as

$$V^{\text{soft}}_{\pi_\theta}(s) = \underset{a \in A_s}{\text{softmax}}\, Q^{\text{soft}}_{\pi_\theta}(s, a), \tag{2.24}$$

$$Q^{\text{soft}}_{\pi_\theta}(s, a) = \theta^T \phi(s, a) + \mathop{\mathbb{E}}_{s' \sim P_{s,a}} \left[V^{\text{soft}}_{\pi_\theta}(s') \right] \tag{2.25}$$

where $\text{softmax}_{x \in X}\, f(x) \triangleq \log \sum_{x \in X} e^{f(x)}$ is a smooth approximation of the $\max_{x \in X}$ operator. The corresponding stochastic policy is given succinctly as

$$\pi_\theta(a|s) = e^{Q^{\text{soft}}_{\pi_\theta}(s,a) - V^{\text{soft}}_{\pi_\theta}(s)} \tag{2.26}$$

Given this model of noisily optimal agent behavior, solving IRL amounts to finding reward parameters θ^* such that the probability distribution induced by π_{θ^*} results in equality between $\hat{\phi}(\theta^*)$ and $\phi_{\mathcal{D}}$. The authors of [5] find this optimal θ^* by maximizing the likelihood of the trajectories in the demonstration set $\mathcal{D}$, subject to respecting the maximum entropy form given in (2.22):

$$\theta^* = \underset{\theta \in \mathbb{R}^{n_\phi}}{\text{argmax}} \sum_{\xi \in \mathcal{D}} \log(P(\xi|\theta)). \tag{2.27}$$

[4] This equivalence is only exactly true for MDPs with deterministic state transitions. As discussed in [5], it serves as an approximation for MDPs with stochastic transitions if agent actions are not greatly influenced by stochasticity (i.e. they employ approximately open-loop policies). Fully addressing stochastic transitions requires the introduction of maximum *causal* entropy [53], [54].

[5] The soft value functions as defined by (2.24) and (2.25) actually relate to the maximum *causal* entropy distribution over trajectories. However, we present them in this form because the causal and non-causal entropy distributions are equivalent for MDPs with deterministic transitions, and this form presents elegant parallelism with the traditional definition of the value functions. The only modification to approximate maximum entropy with stochastic transistions is to switch an expectation and a logarithm operator in (2.25) to obtain $Q^{\text{soft}}_{\pi_\theta}(s, a) = \log \mathbb{E}_{s' \sim P_{s,a}} \left[e^{\theta^T \phi(s,a) + V^{\text{soft}}_{\pi_\theta}(s')} \right]$.

They show that the gradient of this objective at a particular θ is given by the difference between $\phi_{\mathcal{D}}$ and $\hat{\phi}(\theta)$. The optimization can be solved through an iterative gradient decent process: 1. take a gradient step on θ, 2. solve for the new $\hat{\phi}(\theta)$ by computing π_θ through a modified version of value iteration [55] on $V^{\text{soft}}_{\pi_\theta}$ and $Q^{\text{soft}}_{\pi_\theta}$, and 3. repeat these steps until convergence.

While we focus on learning about safety constraints in this dissertation, rather than on learning reward functions, we do still rely on models that can explain human behavior. In particular, our approach to Maximum Likelihood Constraint inference in Chapter 6 relies on the assumption that human actions can be modeled via the policy in (2.26), with the resulting distribution over trajectories given by (2.23). As we discuss in that chapter, by assuming we have access to a nominal reward motivating human behavior, this noisily optimal behavior model provides us with an expected probability distribution over trajectories. When the empirical distribution of trajectories in a demonstration set does not match this expectation, this behavior model provides us with a principled way to infer constraints that will modify our expectations to align them with the available demonstrations.

Notes on Terminology

Often, a given system can be well modeled as either a dynamical system, as in Section 2.1.1, or as an MDP, as in Section 2.3.1, depending on which types of techniques and analyses are of interest. Because we draw on developments from both areas, we employ both models in this dissertation and endeavor to make use of them clearly and consistently. In order to clarify usage, we discuss the relationship between their terminology below.

We first highlight that the dynamical system state x and the MDP state s both capture the same entity, which is the state of the system in its environment. Moreover, dynamical system inputs u are analogous to MDP actions a in how they influence the evolution of the system's state. This evolution is captured by either the dynamics function f or the MDP transition probabilities $P_{s,a}$. Stochasticity in the evolution of the state can be captured directly by these transition probabilities for MDPs, or by an unknown disturbance d entering the dynamics function as in (2.1).

There are, however, some concepts that differ slightly between the two domains. In the context of MDPs, a trajectory ξ typically represents a discrete set of (potentially continuous-valued) states and actions over time. For a dynamical system, a trajectory ζ represents a mapping from time to the state of the system. The time history of the system inputs is usually given separately as the input signal $\mathbf{u}$. The notion of a state's "value" is typically denoted by V in both contexts. Value, in both domains, represents how well an agent (for MDPs) or a controller (for dynamical systems) is expected to perform beginning from a given state; however, the measures of performance can differ. In an MDP, performance is typically measured according to an expected accumulation of rewards, as in (2.13), and a similar formulation is possible for designing optimal control of dynamical systems. However, the meaning of value employed in safety-focused reachability theory for dynamical systems

is based on the minimum distance to a keep-out set, as in (2.4). Whenever we use the term "value" in this dissertation, the local context should make clear which definition we intend.

The definitions and notation that we use throughout this dissertation should generally align with the common usages that we present in this chapter. Occasionally, we need to make slight modifications to facilitate the presentation of our theory, but in such cases we add comments and context that clarify our usage.

Part II

Assisting Safely

Chapter 3

Haptic Assistance via Inverse Reinforcement Learning

As we continue to make advances in the field of robotics, the number of opportunities for people to interact with autonomous and semi-autonomous systems will only continue to increase. In this chapter, we address the challenge of making these interactions as seamless and natural as possible for the human users to improve system performance and safety. Specifically, we use the framework of shared control, detailed in Section 2.2, to examine the role that haptic feedback can play in shaping these experiences.

Haptic shared control has been shown effective in assisted driving by modifying torques or resistances on steering wheels and accelerator peddles [9]–[11] as well as by providing guidance in surgical tasks [43], [44]. In cases where haptic feedback is applied, the feedback policy is typically explicitly specified by hand to reflect expert knowledge of the task, such as generating forces to guide the system towards goal areas (such as desired steering angles) and away from obstacles (such as excessive acceleration). In this work, we make use of Inverse Reinforcement Learning (IRL) to develop haptic feedback policies that implicitly incorporate expert behaviors. As described in Section 2.3, IRL uses a set of observed trajectories to infer the underlying reward function that motivated those trajectories [1]. Using this learned reward function, it is possible to implement a feedback policy for environments where the optimal behaviors are not easily specified.

In this chapter, we use a 2D driving task as our motivating example. We detail our method for using IRL to generate underlying assistance polices, and we demonstrate how those policies can be used to drive both input-mixing and haptic shared control schemes. In Chapter 4, we use those control schemes as test conditions in a user study where we analyze the trade-offs between these two modalities, demonstrating how task performance and user preference vary depending on the setting in which the assistance is applied. Our results provide insight into how haptic feedback affects the user experience of shared control systems.

The work in this chapter first appeared in "Haptic Assistance via Inverse Reinforcement Learning" [13].

3.1 Related Work

3.1.1 Shared Control

Previous work on shared control has successfully used an autonomous agent to assist users in a variety of tasks, including the control of robotic arms [26], [56], [57], drone flight [20], [29], and assisted surgery [40], [43], [44]. While there are a number or approaches for implementing shared control, they generally share the common elements of *predicting* the human's intent and *assisting* to help realize the human's intent [26]. Prediction can be achieved online by continuously monitoring the user's input [12], [43] or a priori by making general assumptions about the user's desires (such as collision avoidance) [20], [29]. In this work, we generate a priori predictions by performing IRL using expert demonstrations and assuming that the user intends to behave similarly to the expert.

Assistance to the user can be provided in a number of ways, as discussed in Section 2.2. For example, *Artificial Force Fields* are based on an underlying control policy that pushes the system towards goals and away from obstacles [29], [37]. *Virtual Fixtures* restrict or impede inputs along certain directions, which can guide users along paths or prevent them from entering dangerous areas [28], [40], [44]. In input-mixing shared control, where assistance is achieved by directly modifying the input signal, these underlying assistance policies are generally combined with the user's input in one of two broad patterns. With a *blending* approach, the actual input to the system is a weighted combination of the user's input and the assistance policy, where the autonomous agent determines how much weight should be given to these two components [12]. As Trautman [42] points out, it is possible for such an approach to fail even when the uncombined inputs would have succeeded individually. On the contrary, in an *inference*-based approach, the control action suggested by the assistance policy is not only based on the current state of the system but also the current input of the human [26], [58].

At first glance, the shared control literature may seem to produce some contradictions, cases where users prefer more [59] or less [60] assistance from the automation. Dragan and Srinivasa reconcile these results by framing a broad spectrum of assistive teleoperation algorithms as instances of their policy-blending formalism [12]. Through their experimentation, they reveal that user preferences for assistance depend on the difficulty of the task being performed and on the accuracy of the automation's prediction about the user's intent. We capitalize on this realization to similarly categorize those cases in which a haptic approach to shared control is preferable to an input-mixing approach.

3.1.2 Haptic Feedback

As we discuss in Section 2.2.3, haptic feedback can serve as a natural communication channel between the human and the automated assistance. We focus specifically on kinesthetic feedback, forces or torques applied by the automation to the human's control interface, which has been applied in scenarios from assisted driving [9]–[11] to assisted surgery [43],

[44]. In previous work, haptic assistance policies have been designed by hand, sometimes employing haptic versions of artificial force fields [11], [29] or virtual fixtures [40], [44] where the parameters of these elements are manually specified. Creating these assistance policies requires expert knowledge of the task on the part of the designer, understanding which behaviors are desirable, what outcomes should be avoided, and knowing the proper balance between these potentially competing objectives. As we mention throughout this dissertation, enabling more advanced automation capabilities will require these systems to rely more on learning and less on human-designed specifications. We look to bring this type of learning to the creation of haptic assistance policies, which motivates us to present our method for generating haptic assistance via inverse reinforcement learning.

3.1.3 Inverse Reinforcement Learning

As we discuss in Section 2.3, in the field of reinforcement learning (RL), encoding a task is typically done via a reward function, which maps state-action pairs to scalar values measuring their relative desirability. Using this reward function, one can measure the performance of a control policy through its induced *value function*, that is, the expected sum of discounted rewards, as given in (2.13). This mapping from states to values encodes how well an agent following a given policy is expected to perform given their current state. The goal of RL is to use observations of the reward to learn a policy that maximizes the value of every state.

The reward function can sometimes be easily encoded with a specific task in mind; however, it is often the case that the task is too complex for the manual design of such a function to be feasible [47], [48]. In those cases, if an expert is available, it is possible to use the expert's behavior as a means to derive the reward structure. This problem of inferring the reward from observed behavior is known as the Inverse Optimal Control (IOC) problem [46] or Inverse Reinforcement Learning (IRL) problem [1].

IRL has been a central topic of AI research for the past two decades, starting with the work of Russell [61] and Ng [1]. Later on, Abbeel *et al.* [62] demonstrated the use of IRL in the context of learning control policies for complicated helicopter maneuvers, and IRL has been applied to domains from path planning [2], [5] to quadruped [4] and humanoid [63] locomotion. There are many techniques available to perform IRL, and we discuss some prominent examples in Section 2.3.2. Inga *et al.* [64] applied IRL in order to understand individual human behaviors in a 2D driving task. They demonstrated that they were able to explain the observed differences in driving trajectories by learning user-specific reward functions. In this work, we are similarly interested in using IRL to infer the implicit reward function of a human navigating a 2D environment. However, not only do we use this learned information to model observed behavior, we also make use of the learned reward to generate an assistance policy. We deploy this IRL-based assistance policy in a shared control setting with the goal of assisting a novice user to follow trajectories that are consistent with expert behavior.

3.2 Haptic Assistance in IRL-based Shared Control

Consider a discrete-time dynamical system under shared control given by

$$\begin{aligned} x_{t+1} &= f(x_t, u), \\ u &= g(u_H, u_A), \\ x \in \mathcal{X} \subseteq \mathbb{R}^n, \quad u, u_H, u_A &\in \mathcal{U} \subseteq \mathbb{R}^{n_u}, \end{aligned} \tag{3.1}$$

where x_t and x_{t+1} are the state of the system at time $t \in \mathbb{Z}$ and the next discrete time, respectively, and the system input, u, is a function of u_H, the input provided by the human user, and u_A, the input selected by the assistance policy. The evolutions of these values over time are given by a state trajectory $\mathbf{x} : \mathbb{Z} \to \mathcal{X}$ and input signals $\mathbf{u}, \mathbf{u}_H, \mathbf{u}_A : \mathbb{Z} \to \mathcal{U}$. This system model combines notation typical for discussing the control of dynamical systems (as in (2.1)), with the discrete-time nature commonly used to describe MDPs and perform IRL (as in Section 2.3).

With this discrete-time form, we can define the value function V_π for this system as the discounted sum of rewards generated by following a control policy $\pi : \mathcal{X} \to \mathcal{U}$, as in (2.13). The *optimal* value function V^* is the value corresponding to a control policy π^* that maximizes the value at every state. Given the optimal value function V^* for a given task, the optimal input u^* at any time t can be selected by maximizing the sum of the value at the next time step, plus the reward earned at the current time step. If we assume that providing any input is no more burdensome to the human than any other, then the reward depends only on the state, and the optimal input at state x_t can be chosen by solving

$$\begin{aligned} u^*_{x_t} = \underset{u \in \mathcal{U}}{\operatorname{argmax}} \; & V^*(x_{t+1}), \\ \text{s.t. } & x_{t+1} = f(x_t, u). \end{aligned} \tag{3.2}$$

In an input-mixing, blending approach to shared control, the autonomous assistance will select $\mathbf{u}_A(t) = \mathbf{u}^*(t)$ along with a, potentially time-varying, blending parameter $\alpha : \mathbb{Z} \to [0, 1]$ such that

$$\mathbf{u}(t) = \alpha(t)\mathbf{u}_A(t) + (1 - \alpha(t))\mathbf{u}_H(t). \tag{3.3}$$

The blending parameter α trades off the amount of control authority allotted to the human or the assistance, and the value of α could represent, as in [12], a measure of the assistance's confidence in its prediction about the user's goal. That is, α can correspond to the likelihood that $V^*(\cdot)$ is the correct value function to describe the human's intended control task.

While shared control of this form will ideally properly take into account the human's actions to blend the two input sources, it is possible for the automated assistance to completely override a human's input by choosing $\alpha(t) = 1$ for all t. Even without taking full control authority away from the human, if $\mathbf{u}_A$ significantly deviates from $\mathbf{u}_H$, then it is possible for the system to follow trajectories that deviate significantly from those that would have

occurred if the human had been unassisted [42]. Ultimate control of the system lies with the automation. Although it may be desirable to apply significant corrections to user input if the assistance correctly predicts the user's goal, it can lead to frustration if the incorrect goal is predicted [12].

We put forth that the benefits of replacing input mixing with haptic feedback shared control can be understood as reducing this frustration by returning the ultimate control of the system to the user: haptic shared control simply lets $\mathbf{u} = \mathbf{u}_H$ and uses haptic feedback to influence the user's selection of $\mathbf{u}_H$. As long as the user can overpower the haptic assistance forces, the user has the ultimate authority to select the system input signal $\mathbf{u}$. The responsibility of intelligently blending inputs is shifted from the automated assistance to the human: rather than the automation selecting a value for α, the human selects how stiff or compliant to be in response to the assistance forces.

In what follows, we detail our approach for constructing haptic assistance policies for shared control. First, in Section 3.2.1, we detail our approach to using IRL for generating underlying value functions from expert demonstration. Then, in Section 3.2.2, we explain how we leverage these value functions to implement haptic shared control. Finally, in Chapter 4, we present a user study performed to evaluate the effects of using haptic feedback in lieu of input mixing given the same underlying IRL-based assistance policy.

3.2.1 Inverse Reinforcement Learning for Navigation Assistance

In order to provide assistance for shared control, we seek a policy $\pi^*(x) = u_x^*$, as in (3.2), based on the optimal value function $V^*(\cdot)$. Before we can find this optimal value function, however, we must first learn the underlying reward function of the environment, $R(\cdot)$. An estimate of this reward function can be generated by applying an IRL algorithm to expert trajectories. Given the learned reward function, the optimal value function can be found through methods such as generalized policy iteration [65] or estimated via more complex RL algorithms.

As mentioned above, we are specifically motivated by the case of 2D navigation, which is the task that humans must solve to maneuver vehicles along roadways. Given this setting, we present our approach to the general problem given in the preceding paragraph. We describe positions in the 2D environment by two orthogonal coordinates $[x_1\ x_2]^T \in \mathcal{X}$. The vehicle we focus on will move forward with a constant speed v, but its direction of motion is determined by an angular input u, such that its continuous time dynamics are given by

$$\begin{aligned} \dot{x}_1 &= v\cos(u), \\ \dot{x}_2 &= v\sin(u). \end{aligned} \tag{3.4}$$

When performing IRL, we assume that the observed trajectories are generated by an *unassisted* expert such that $u = u_H$. It is straightforward to discretize these dynamics with respect to time to recover a discrete-time dynamical system, as in (3.1).

In order to prepare expert trajectories for max-margin IRL, we divide the state space into a coarse grid of equally sized tiles, S, with each tile $s \in S$ centered at $x_s \in \mathcal{X}$ and

encompassing all states $x \in \mathcal{X}$ such that $\|x - x_s\|_\infty < \delta$, where $\delta \in \mathbb{R}$ is the coarseness of the grid. Letting these state tiles themselves constitute the features of the environment, expert trajectories of length T can be represented as a T-tuple of visited tiles $\phi = (s_0, s_1, \ldots, s_T)$. The set of actions over this tile space can be given as $A = \{up, down, left, right\}$, where each action causes a transition from a tile to one of its four adjacent tiles. We choose a reward function $R : S \to \mathbb{R}$ that only depends on the current tile, such that it can be encoded as a vector $\mathbf{R} \in \mathbb{R}^{|S|}$. Correspondingly, we can represent a trajectory's feature history ϕ by a matrix $\boldsymbol{\phi} \in \{0,1\}^{T \times |S|}$, such that the ts-th element is zero unless the trajectory visited tile s at time t. If we additionally allow $\Gamma' = [1 \ \gamma \ \gamma^2 \ ... \ \gamma^{T-1}]$ to be a row vector of discounts over time, then the discounted sum of rewards earned over a trajectory is given by $\Gamma' \boldsymbol{\phi} \mathbf{R}$.

As previously stated, the IRL approach we will be using to estimate the reward will be a *max-margin* approach, as in [2], [3], based on a standard optimization of the form:

$$\begin{aligned} \underset{||\mathbf{R}||_2 \leq 1}{\text{maximize}} \quad & l \\ \text{subject to} \quad & \frac{1}{N}\Gamma'\Big(\sum_{i=1}^{N} \boldsymbol{\phi}_i^E - \boldsymbol{\phi}_i^j\Big)\mathbf{R} \geq l + \lambda \Big|\Big| \sum_{i=1}^{N} \boldsymbol{\phi}_i^E - \boldsymbol{\phi}_i^j \Big|\Big|_1, \\ & j = 0, \ldots, m. \end{aligned} \tag{3.5}$$

where N corresponds to the number of sampled trajectories, and $\boldsymbol{\phi}_i^E, \boldsymbol{\phi}_i^j \in \{0,1\}^{T \times |S|}$ represent the sequence of states visited by the i-th trajectory from the expert and from the j-th estimated policy, respectively (see Algorithm 3.1). Here, λ is a small constant that penalizes trajectories that are far from the expert's.

The optimization problem in (3.5) can be interpreted as trying to maximize the margin between the expert's (sampled) mean performance and the policies', while penalizing those policies whose trajectories differ from the expert's. With this optimization procedure in place, Algorithm 3.1 was used to obtain the assistance policy.

In Algorithm 3.1, *CollectExpertTrajectories* and *CollectPolicyTrajectories* are functions used to obtain N rollouts from the expert and the intermediate policies from a predefined set of starting states. *ValueIteration* runs the well-known value iteration algorithm [66] up to an ϵ accuracy in order to obtain the policy and value function. For the deterministic transition function, off road states and goal states were taken to be trapping states.

Even though we are using a max-margin IRL approach to compute the reward function and value iteration to obtain the value function, we note that the algorithm can be implemented with any desired IRL / RL combination to compute the reward and value functions.

Finally, it is important to note that the algorithm just described returns a discrete value function over the set of state space tiles S. Recall, however, that in order to provide haptic assistance for shared control, we will need a value function over the continuous state space $\mathcal{X}$. We construct such a continuous function $V^* : \mathcal{X} \to \mathbb{R}$ by letting $V^*(x_s) = V^*(s)$ for all tile center points and performing multilinear interpolation to evaluate $V^*(x)$ for any point x which does not lie at the center of a tile. In the following sections, V^* will refer to this continuous value function.

Algorithm 3.1 IRL for assistance policy synthesis

Input: N, M, λ, γ
Output: $\mathbf{R}$, V^*, π^*

$\pi_{rnd} \leftarrow RandomPolicy()$
$\boldsymbol{\phi}^E \leftarrow CollectExpertTrajectories(N)$
$\boldsymbol{\phi}^0 \leftarrow CollectPolicyTrajectories(N, \pi_{rnd})$
$\Phi_{set} \leftarrow \{\boldsymbol{\phi}^*, \boldsymbol{\phi}^0\}$
for $m = 0$ to M **do**
  $\mathbf{R} \leftarrow Solve$ (3.5) $with$ Φ_{set}
  $V^*, \pi^* \leftarrow ValueIteration(\mathbf{R}, \epsilon)$
  $\boldsymbol{\phi}^{m+1} \leftarrow CollectPolicyTrajectories(N, \pi^*)$
  $\Phi_{set} \leftarrow \Phi_{set} \cup \{\boldsymbol{\phi}^{m+1}\}$
end for
Return $\mathbf{R}, V^*, \pi^*$

3.2.2 Haptic Shared Control from Value Functions

Once the value function V^* has been obtained, a straightforward approach to haptic assistance is simply to select u_A to be the global optimum according to (3.2). Because the automated assistance believes that u_A is the best possible input given the current state of the system, the goal of haptic feedback is to influence the human user to align their input with u_A. In order to achieve this goal, we adopt the model of a saturated, multi-dimensional, damped spring:

$$F(t) = \sigma\Big(- K\big(\mathbf{u}_H(t) - \mathbf{u}_A(t)\big) - B\big(\mathbf{u}_H(t) - \mathbf{u}_H(t-1)\big)\Big), \tag{3.6}$$

where $\sigma : \mathbb{R}^{n_u} \rightarrow \mathbb{R}^{n_u}$ is an element-wise saturation function, $K, B \in \mathbb{R}^{n_u \times n_u}$ are diagonal matrices with positive entries, and $F(t) \in \mathbb{R}^{n_u}$ is a vector of forces and/or torques (depending on whether the human's input interface is rotational or translational for each input dimension) applied by the automation to the input interface at time t. The saturation function $\sigma(\cdot)$ ensures that these forces and torques are bounded, which is an important consideration due to the actuation limitations of haptic interfaces and safety, comfort, and usability concerns for human users. For simplicity, we will use the term "force" to refer to both forces and torques moving forward.

In the case where u_A takes a constant value (e.g. u^0: the zero vector for all time), by applying the haptic feedback policy given by (3.6), the user will always feel a restorative force pulling them towards the equilibrium point u^0. In the more general case where u_A varies over time, a user will feel as if their input interface is being pulled by a spring attached to the moving value of u_A. This virtual spring force is equivalent to a time-varying virtual force field which attracts to the current value of u_A.

It is important to note that, unlike input mixing shared control, which can instantaneously change the system input u, haptic shared control is only able to change u by in-

fluencing u_H through forces applied to the user interface. This relationship leads to two fundamental properties of haptic shared control:

- Haptic assistance must be applied *anticipatively*.
- Haptic shared control is *restricted* in the level of assistance that it can provide.

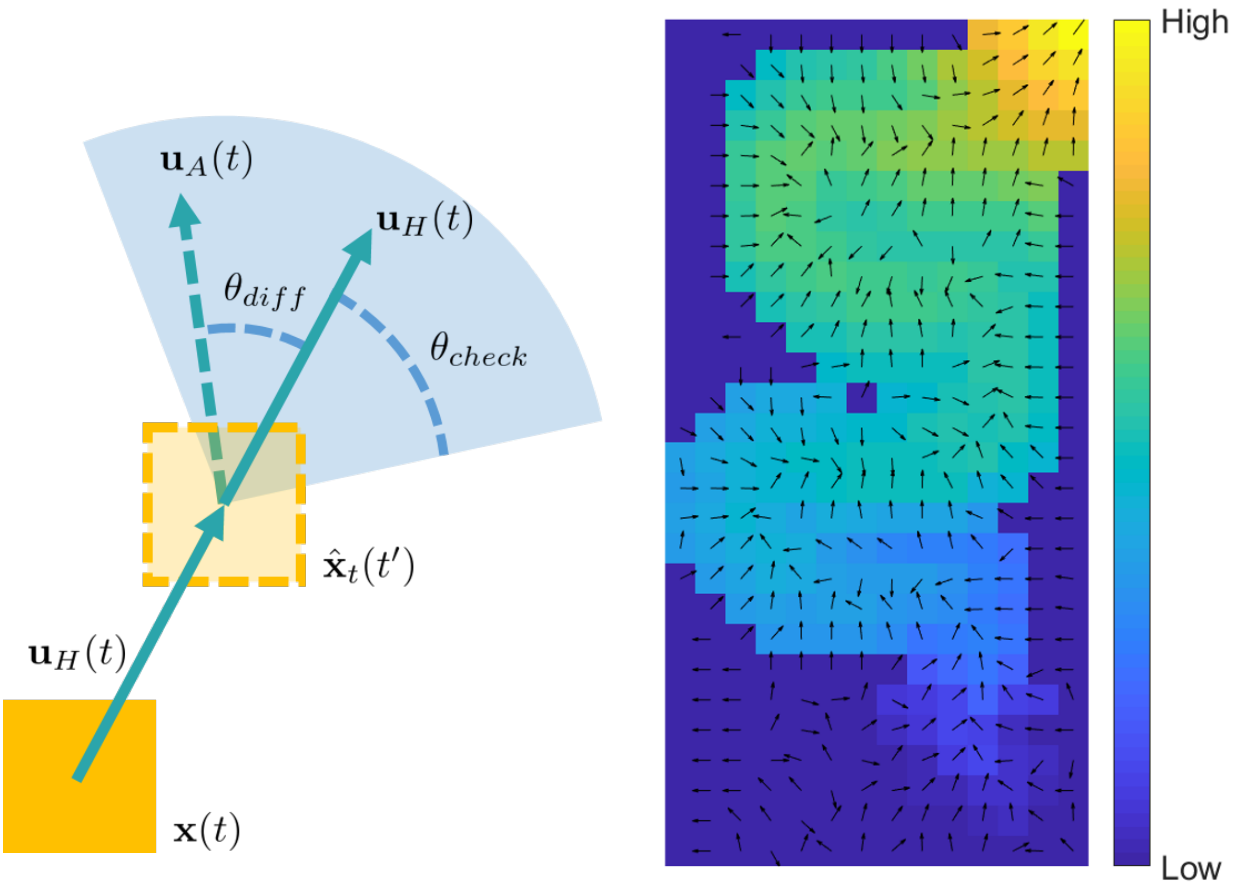
The requirement of anticipative haptic assistance arises from the facts that humans are not able to instantaneously react to forces and input interfaces are not able to instantaneously travel from one configuration to another. Similarly to [10], preliminary trials of the experiment in Chapter 4 revealed that if $F(t)$ is based on a value of $\mathbf{u}_A(t)$ computed as in (3.2), that is, based directly on the system configuration at time t, then the user may perceive these forces as lagging behind the system: by the time the user can react to these forces, they no longer relate to the current state of the system. To combat this issue, haptic assistance should be based on some form of Look-Ahead Guidance as in [10]. For our approach, this means that $\mathbf{u}_A(t)$ should ideally be selected based on the future state $\mathbf{x}(t')$ of the system at some future time $t' = t+T'$, where T' captures the reaction time of the user to the haptic assistance. Assistance applied in this manner will elicit responses from the user at the appropriate time to be effective. Because shared control systems are causal, we must form an estimate of the future state denoted as $\hat{\mathbf{x}}_t(t')$, the state of the system at time t' as predicted at time t. The selection of $\mathbf{u}_A(t)$ is then given by the following:

$$\begin{aligned} \mathbf{u}_A(t) = \underset{u \in \mathcal{U}}{\operatorname{argmax}}\ & V^*(\hat{\mathbf{x}}_t(t'+1)), \\ & \text{s.t. } \hat{\mathbf{x}}_t(t'+1) = f(\hat{\mathbf{x}}_t(t'), u), \\ & \quad t' = t + T'. \end{aligned} \tag{3.7}$$

Following from the fact that haptic assistance acts on a limited time scale and with bounded forces, haptic shared control is restricted in its level of assistance. That is, selecting u_A according to (3.7) is still insufficient for a haptic assistance approach: $\mathbf{u}_A(t)$ may be far removed from the current $\mathbf{u}_H(t)$, such that the assistance force will not be able to drive the user's input sufficiently towards $\mathbf{u}_A(t)$ over a relevant time period. Given this limitation, better performance could be achieved by selecting the optimal input that lies within a neighborhood of the current $\mathbf{u}_H(t)$. We thus reformulate the selection of $\mathbf{u}_A(t)$ as:

$$\begin{aligned} \mathbf{u}_A(t) = \underset{u \in \mathcal{U}}{\operatorname{argmax}}\ & V^*(\hat{\mathbf{x}}_t(t'+1)), \\ & \text{s.t. } \hat{\mathbf{x}}_t(t'+1) = f(\hat{\mathbf{x}}_t(t'), u), \\ & \quad t' = t + T', \\ & \quad \|u - \mathbf{u}_H(t)\| \leq \delta_u, \end{aligned} \tag{3.8}$$

where $\delta_u \in \mathbb{R}_+$ is chosen to restrict the values of $\mathbf{u}_A(t)$ under consideration to be input values that are achievable with bounded haptic assistance.



(a) $\mathbf{u}_A(t)$ selection

(b) Learned value function

Figure 3.1: The primary components of our assistance approach, as applied to our user study in Chapter 4. Following (3.8), and shown in (a), selecting $\mathbf{u}_A(t)$ is done by first predicting the future state assuming constant input, then searching to find the optimal input within an angular arc of $2\theta_{check}$ about $\mathbf{u}_H(t)$. The optimality of potential inputs is judged according to a value function generated via IRL, shown in (b) for the Hard road in our study (see Figure 4.1c). The arrows indicate the interpolated direction of greatest value increase at each point.

We adopt the input selection in (3.8) and the force calculation in (3.6) as our approach for translating learned value functions into actionable haptic assistance policies. Figure 3.1 shows an example of this assistance method as applied in the user study detailed in Chapter 4.

Chapter 4

Improving Shared Control with Haptic Feedback

4.1 Study on Haptic Assistance Preferences

In order to test our ideas about the effects of haptic feedback on shared control interfaces, detailed in Chapter 3, we conducted an experiment where subjects used a haptic feedback device (Phantom Premium 1.5 6DOF [45]) to carry out a computer simulated control task. The users would hold the handle of the device and they would control the simulation shown in Figure 4.1. The yellow vehicle moved at a constant speed, but with a variable heading based on the user's input, u_H, which corresponded to the angular position of the device handle (represented in simulation by a teal arrow, as in Figures 4.1 and 3.1a). Subjects were tasked with driving the vehicle from the bottom of the map to the top while remaining on the road.

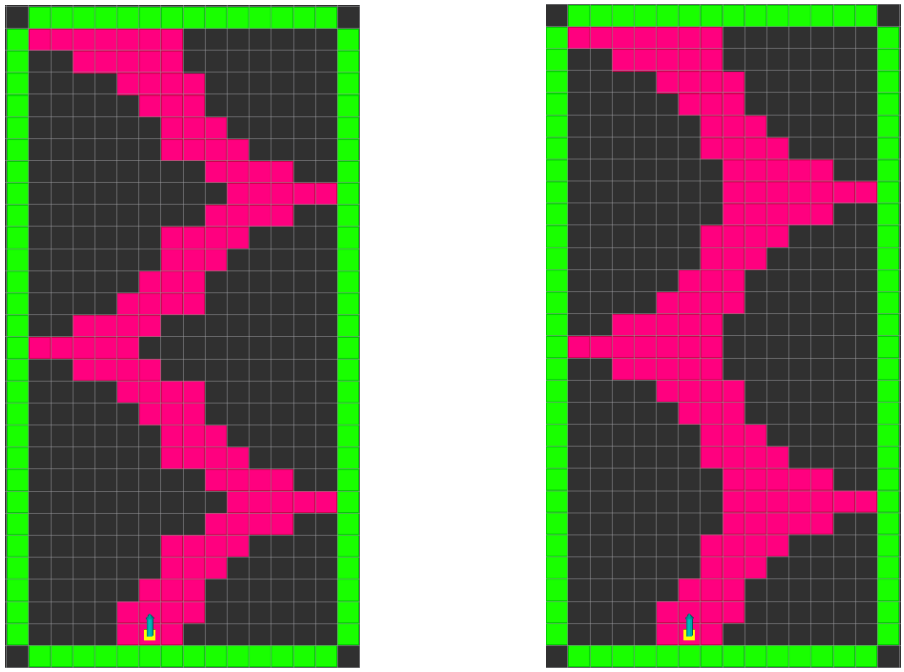
4.1.1 Roadmaps

We created two primary road maps that we would ask subjects to navigate, an Easy and a Hard road, shown in Figures 4.1a and 4.1c. In order to successfully steer the vehicle along the Easy road, the user only needs to rotate their wrist by approximately 45°, well within a typical person's comfortable workspace. On the contrary, the Hard road requires wrist rotation of approximately 90°, which approaches the edge of the typical human workspace [67].

4.1.2 Assistance Policies

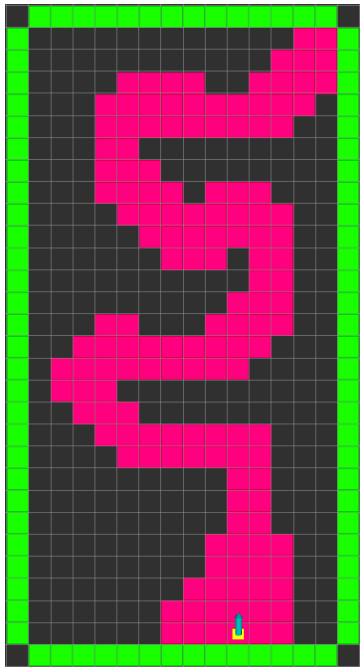
To generate the value functions that would underlie our assistance policies, we performed the IRL-based procedure detailed in Section 3.2.1 using expert trajectory data gathered from one of the authors successfully performing the task, with an example result shown in

The work in this chapter first appeared in "Haptic Assistance via Inverse Reinforcement Learning" [13].

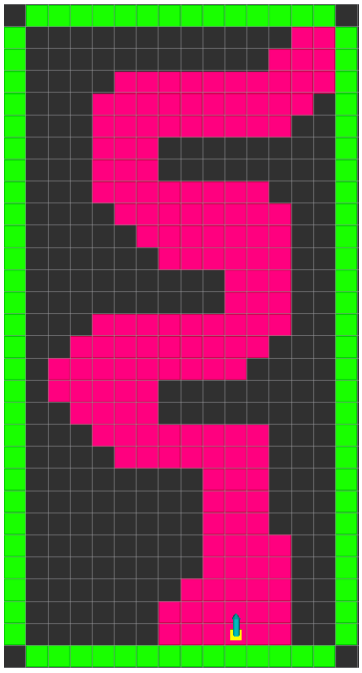


(a) Easy

(b) Easy (Wrong)



(c) Hard



(d) Hard (Wrong)

Figure 4.1: The roadmaps employed in the user study. Navigating the Easy road (a) requires less wrist rotation than the Hard road (c). The Wrong variants, (b) and (d), are designed to induce navigation assistance that is generally but imperfectly aligned with the original roads.

Figure 3.1b. For both the Easy and the Hard road, trajectories were gathered for the roads as shown in Figures 4.1a and 4.1c, as well as modified versions that expanded the width of the road in certain locations, shown in Figures 4.1b and 4.1d. Value functions generated from the original roads correspond to Right assistance predictions (they describe the actual road on which the user will drive) while value functions generated from the modified roads correspond to Wrong predictions (not perfectly aligned with the navigation goal).

We created both an input-mixing- and a haptic-feedback-based approach to assistance, which we dub IM and HF, respectively. HF is created by selecting $\mathbf{u}_A(t)$ as in (3.8) and generating torques on the haptic device handle as in (3.6). To implement the input bound condition from (3.8), we restricted the search for inputs to a range of $\pm\theta_{check} = \pi/8$, as shown in Fig 3.1a. For the purpose of comparison in this study, we apply the same restriction to input selection for IM, however we do not first predict a future state since input-mixing methods are able to instantaneously alter the input to the system. IM performs blending shared control as in (3.3) with $\alpha(t) = 1$ for all t, meaning that the input passed to the system is the optimal input within a neighborhood of $\mathbf{u}_H(t)$.

4.1.3 Experimental Design

We recruited 12 participants for this study from among the graduate student population of our department. While some where familiar with robotics in general, none were familiar with our specific project or the use of haptic interface devices. Of the subjects, 5 were female and 7 were male, and their ages ranged from 20 to 28.

The experimental procedure was divided into four scenarios: the cross of the Easy and Hard roadmaps with the Right and Wrong assistance policies. Every subject experienced each of these four scenarios, but the order in which they were presented to the subjects was counter balanced using a balanced Latin square scheme to combat ordering effects. In each scenario, the subjects were asked to perform the driving task described above 6 times: 3 consecutive trials with each of the two assistance paradigms, IM and HF. The ordering of the paradigms was balanced such that each subject and each scenario was paired with the two possible orderings an equal number of times (twice per subject and six times per scenario).

4.1.4 Measures

In order to objectively evaluate task performance, we recorded how many off-road states the vehicle entered (a measure of safely achieving the control goal) as well as the total angular distance travelled by the haptic device's handle (a proxy measurement for the amount of cognitive effort expended by the user). For each user, we record the average value of these measures from their three trials with each assistance mode.

We also collected subjective data about user preferences. After experiencing both assistance modes for a given scenario, users were asked to use a seven-point Likert scale to respond to a number of statements about each mode of assistance, such as "I found System

[A/B] helpful" or "I felt in control using System [X/Y]." Statements were grouped into three categories measuring how in control users felt, how natural they found the assistance, and how helpful they found the assistance. These categories were found to be internally consistent (Cronbach's $\alpha > 0.85$ for each), and the recorded responses are the average response across statements within a category. Users were also directly asked which mode of assistance they preferred using the same seven-point scale, with 1 and 7 corresponding to extreme preferences for one mode over the other and 4 representing indifference.

4.1.5 Hypotheses

1. There will be a significant interaction between Prediction and Assistance and between Difficulty and Prediction on task performance (IM will perform better for Right predictions, while HF will perform better for Wrong predictions; however, these results will be more pronounced on the Hard road than the Easy road).

2. There will be a significant effect of Prediction on user preference (IM will be preferable with Right predictions, but HF will be preferable for Wrong predictions) and sense of control (HF will give users a greater sense of control with Wrong predictions).

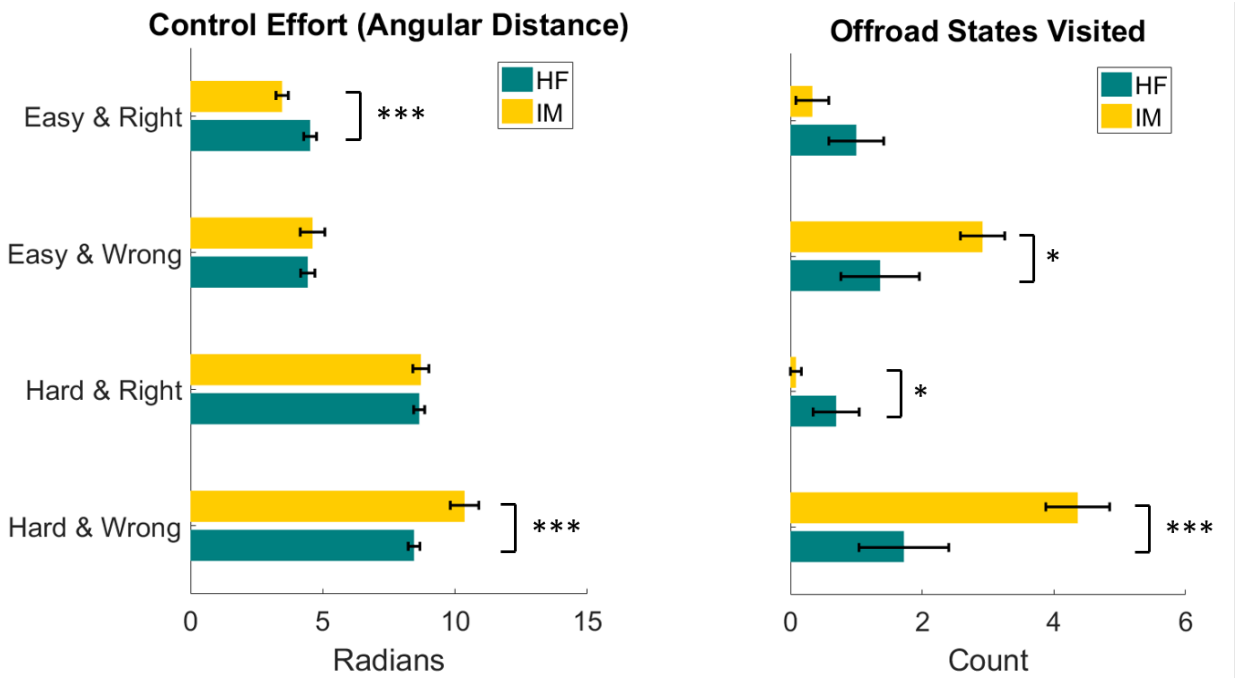
4.2 Analysis and Discussion

4.2.1 Task Performance

We performed a factorial repeated-measures ANOVA on the data with a significance level of $\alpha = 0.05$. For our measure of control effort, significant interactions were identified between Prediction and Assistance ($F(1, 11) = 16.79$, $p < .01$) and between Difficulty and Assistance ($F(1, 11) = 35.93$, $p < .001$). Post-hoc analysis using a Wilcoxon paired signed rank test revealed that while IM required less control effort than HF on the Easy map with Right predictions ($p < .001$), IM required significantly more effort than HF on the Hard map with Wrong predictions ($p < .01$) (Figure 4.2a).

Additionally, for the number of offroad states visited, there were significant interactions between Difficulty and Prediction ($F(1, 11) = 9.21$, $p < .05$) and between Prediction and Assistance ($F(1, 11) = 64.57$, $p < .001$). Examining the data reveals that, when the prediction was Right, IM lead to significantly fewer offroad states visited than HF on the Hard road ($p < .05$). However, when the prediction was Wrong, HF had significantly fewer offroad states visited than IM on both the Easy ($p < .05$) and Hard ($p < .01$) maps (see Figure 4.2b).

Overall, these effects support our first hypothesis. When input-mixing assistance was based on correct predictions of the road, users expended less control effort to successfully navigate the Easy road and exited the Hard road less frequently compared to their performance with haptic feedback. However, given assistance based on an incorrect prediction of the road, users were better able to avoid offroad states under haptic feedback than with input-mixing, and they did so with less control effort on the Hard road.



(a) Control Effort

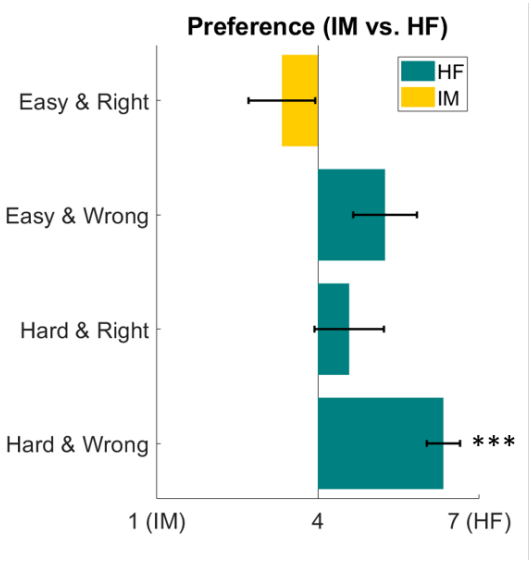
(b) Offroad States Visited

Figure 4.2: Task performance for each of the four scenarios, with error bars representing ± one standard error. IM required significantly less control effort for the Easy and Right case, but HF required significantly less control effort for the Hard and Wrong case. With Wrong predictions, HF lead to significantly fewer offroad states visited, but IM lead to fewer offroad states in the Hard and Right case (* - ($p < .05$), *** - ($p < .001$), based on a Wilcoxon paired signed-rank test).

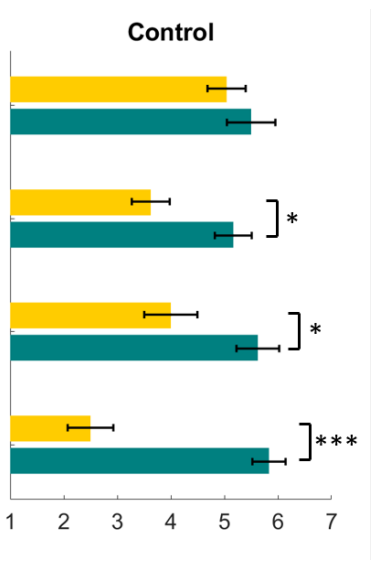
4.2.2 User Preference

We found a significant main effect of Prediction on the reported user preference. With Right predictions, there is no significant preference between HF and IM on either the Easy ($p = .3$) or Hard ($p = .48$) roads. In contrast, when predictions are Wrong, there is a marginally significant preference for HF on the Easy road ($p = .068$) and a strongly significant preference for HF on the Hard road ($p < .001$) (see Figure 4.3a).

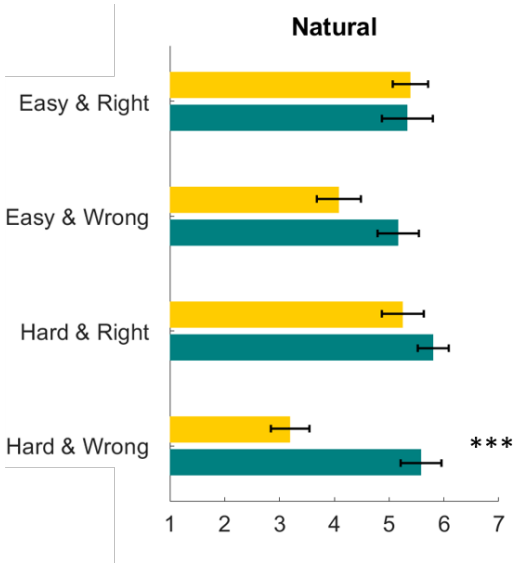
Analysis also revealed a significant interaction of Prediction and Assistance on users' reported sense of control during the task ($F(1, 11) = 10.63$, $p < .01$). In scenarios with Wrong predictions, users reported a significantly greater sense of control with HF ($p < .05$ on Easy, $p < .001$ on Hard), which we suspect to be a major contributor for users' preference for HF in these cases. When the prediction is Right, however, this effect is weaker on the Hard road ($p < .05$), and it is not significant on the Easy road ($p = .37$) (see Figure 4.3b). We suspect that the sense of control with IM in the Easy and Right case is due to an effect reported by Dragan and Srinivasa that occurs when assistance is so well aligned with the



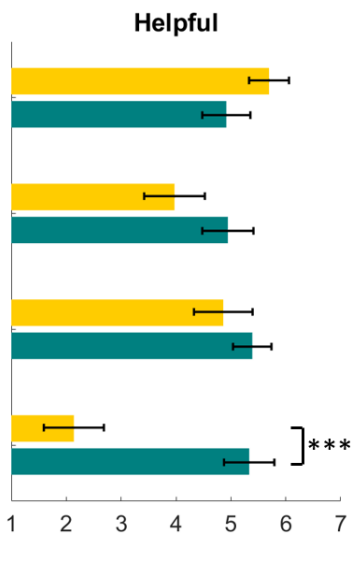
(a) Preference



(b) Sense of Control



(c) Felt Natural



(d) Helpful Assistance

Figure 4.3: Results of the survey given to participants in the user study, where the error bars show ± one standard error. Figs (b) through (d) show the average response value to questions that gauged a user's sense of control, how natural they thought the assistance felt, and how helpful they found the assistance, respectively. There is no significant preference between IM and HF for Right predictions, but there is a strongly significant preference for HF in the Hard and Wrong condition. Additionally, when predictions were Wrong, users felt significantly more in control with HF than with IM (* - ($p < .05$), *** - ($p < .001$), based on a Wilcoxon paired signed-rank test).

user's intent that they do not feel as though they are being assisted [12].

Overall, these results partially support our second hypothesis. While users did not significantly prefer IM with Right predictions, there was a significant preference for HF in the Hard and Wrong case. This preference may be partially explained by the fact that users felt significantly more in control with HF when they were required to override Wrong predictions for assistance.

4.3 Conclusion

In Chapter 3, we detail our approach for using IRL to generate haptic assistance policies to aid users in shared control tasks, and in this chapter we demonstrate that on such tasks, using haptic feedback can mitigate the detrimental effects of incorrectly predicting the user's intent. In particular, we show that haptic feedback can be used to maintain a user's sense of control while still providing assistance, as well as providing a reduced degradation of task performance and safety when the assistance's prediction is incorrect. These results suggest that in situations where predictions are static or there is a reasonable likelihood of making incorrect predictions, haptic feedback could be used to provide assistance in lieu of a timid arbitration policy from an input-mixing system. By shifting the role of arbitration to the human, they are better able to reject erroneous assistance, and they become the final control authority in the human-robot team.

Part III

Learning Safety

Chapter 5

Supervisor Safe Sets

As automation becomes more pervasive throughout society, humans will increasingly find themselves interacting with autonomous and semi-autonomous systems. These interactions have the potential to multiply the productivity of human workers, since it will become possible for a single human to supervise the behavior of multiple robotic agents. For example, a single human driver could manage a fleet of self-driving delivery robots, but the driver would only take full control for the "last mile," guiding the robots to precisely deposit packages in environments where autonomous navigation may not be reliable. Human experts regularly serve as failsafe supervisors on factory assembly floors staffed with robotic arms [68]. Air traffic controllers soon will have to manage completely autonomous drones flying through their airspace alongside existing traditional mixed-autonomy planes and their autopilots [69].

While a human may be able to successfully exert direct control over a single robot, it becomes intractable for a human to directly control teams of robots (in fact, humans often benefit from automated assistance when controlling even a single robot, as discussed in the literature on assistive teleoperation [12], [70]). In order to manage the increased complexity of multi-robot teams, the human must be able to rely on increased autonomy from the robots, freeing the human to focus their attention only on those areas where they are most needed. Our goal is to model what grabs the supervisor's attention in order to modify robot behavior to reduce the occurrence of distractions.

This project is inspired by work like Bajcsy *et al.* [33] and Jain *et al.* [71] that learn from supervisor interventions in a "coactive" learning framework. These works apply Learning from Demonstration techniques to the more challenging domain where the given data is just a correction from a trajectory rather than a full trajectory. The authors of [33] posed this correction challenge in model-based framework that interprets the human's signals as resulting from an optimization problem. This inverse optimization framework has also been used in Inverse Reinforcement Learning [3], [5] which applies Inverse Optimal Control (as conceived of by Kalman [46]) to interpreting human trajectories (see Section 2.3 for more

The work in this chapter first appeared in "Modeling Supervisor Safe Sets for Improving Collaboration in Human-Robot Teams" [14].

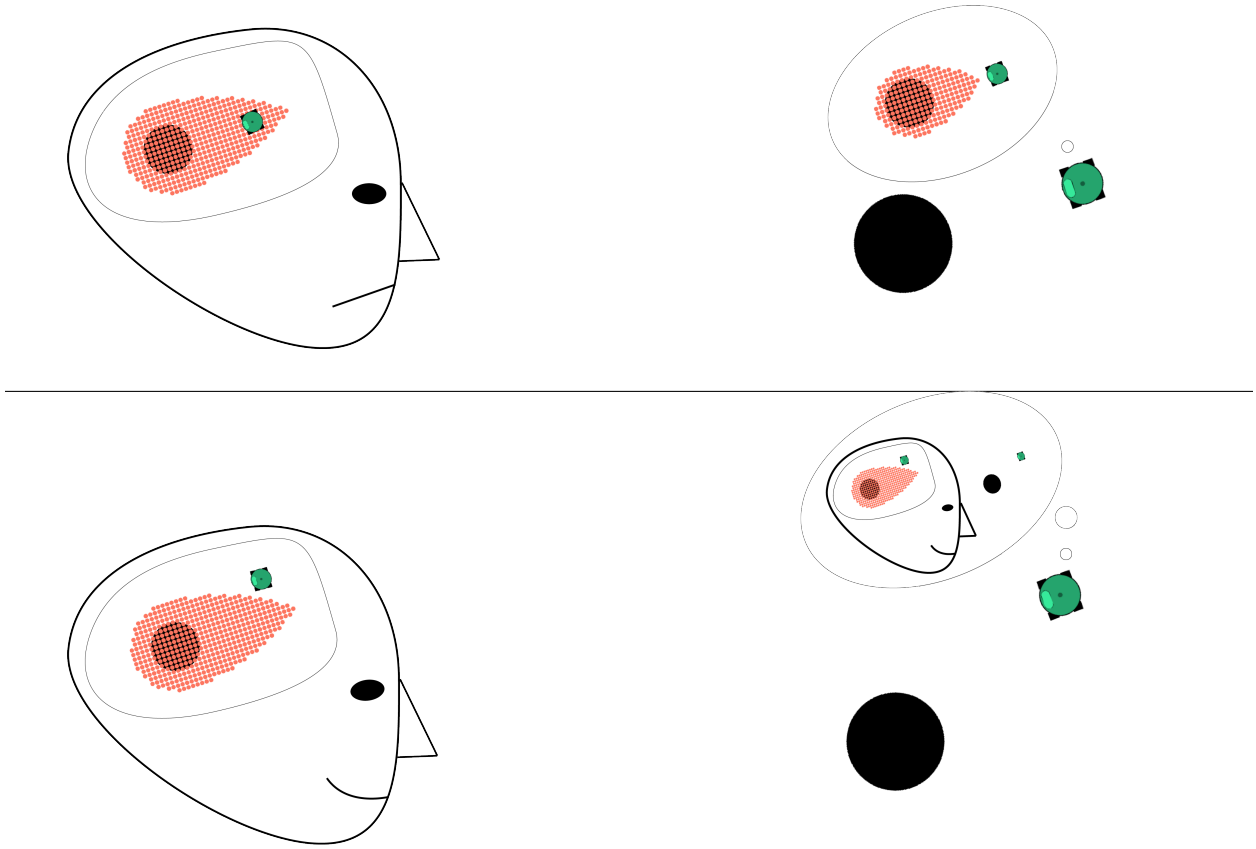


Figure 5.1: Top: if a robot's behavior does not take into account a human supervisor's notion of safety, the misaligned expectations can degrade team performance. Bottom: When a robot acts according to a human supervisor's expectations, the supervisor can more easily predict the robot's behavior.

details). Our work applies the inverse optimization framework to learn from the supervisor's decisions to intervene.

Results in cognitive science suggest that humans observing physical scenes can be modeled as performing a noisy "mental simulation" to predict trajectories [72], [73]. The use of mental dynamical models is also supported by work in neuromechanical motor control which asserts that human action mastery involves building a dynamical model of the human body [74]. Robinson *et al.* posited that this dynamics learning extends not just to direct control of the body but to external systems with which the human interacts, such as human-cyber-physical systems [75]. We posit that human supervisors utilize this same cognitive dynamic simulation to predict robot safety and intervene accordingly. Specifically, we theorize that the intervention behavior is driven by an internal "safe set" which we can attempt to reconstruct by observing supervisor interventions.

Safe sets are conceived from the Formal Methods notion of "Viability". A set of states is

"viable" if for every state in the set there exists a dynamic trajectory that stays within the set for all time. Reachability analysis, as described in Section 2.1, calculates the largest viable set that doesn't include any undesirable state configurations (e.g. collisions with obstacles, power overloads, etc.). Since the set is viable, it is possible to guarantee that the dynamic system will always stay within the set and therefore stay safely away from the undesirable states. For this reason, viability kernels are often referred to as "safe sets". Reachability can be used for robust path planning in dynamically changing environments [76] or working around multiple dynamic agents [77], and recent results have leveraged the technique to bound tracking error in order to generate dynamically feasible paths using simple planning algorithms [78].

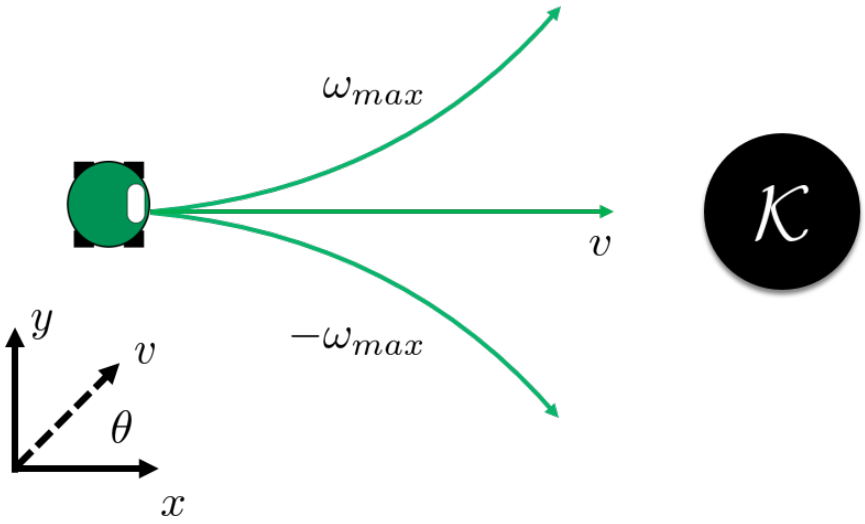
Hoffman *et al.* [79] used the safety guarantees of reachability analysis to engineer a multi-drone team that could automatically avoid collisions. Similarly, Gillula [20] could guarantee safety for learning algorithms by constraining their explorations to stay within the safe set. Extending this, Akametalu and Tomlin [80] were able to guarantee safety while simultaneously learning and expanding the safe set. All of these controllers supervise otherwise un-guaranteed systems and intervene to maintain safety whenever the system threatens to leave the viable safe set. In this paper, we explore how this intervention behavior is similar to human supervision, and apply this to representing human safety concerns as safe sets in the state space.

5.1 Supervisor Safe Set Control

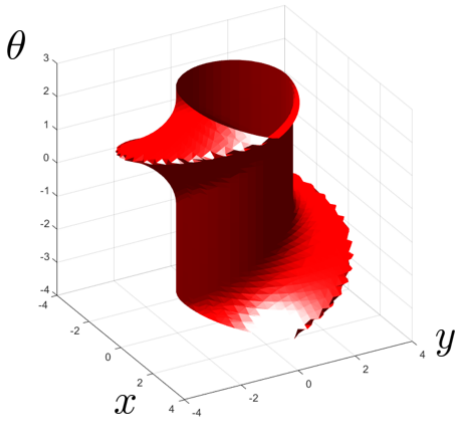
Based on the success of cognitive dynamical models for explaining humans' understanding of physical systems, we posit that human operators may have some notion of reachable sets which they employ to predict collisions or avoid obstacles. We propose a noisy idealized model to describe the behavior of the human supervisor of a robotic team, and we develop a framework for estimating the human supervisor's mental model of a dynamical system based on observing their interactions with the team. We then propose a control framework that capitalizes on this learned information to improve collaboration in such human-robot teams. We describe our ideas using the notation for reachability analysis that we present in Section 2.1, and which we briefly summarize here. To ground our discussion, we will visualize concepts using the Dubins car system described by (5.9) and depicted in Figure 5.2.

5.1.1 Reachability for Safety

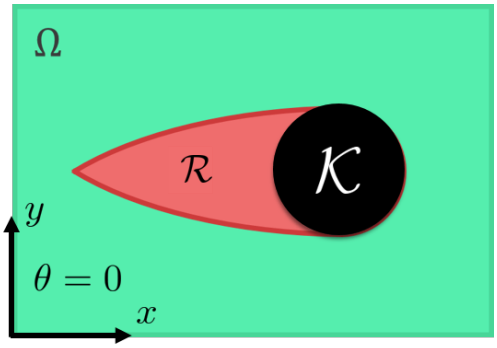
In this formulation, we adopt the notion of a dynamical system described in Section 2.1.1, whereby the evolution of a system's state $x \in \mathbb{R}^n$ is governed by a dynamics equation $f(x, u, d)$, which also depend on an input $u \in \mathcal{U} \subset \mathbb{R}^{n_u}$ and disturbance $d \in \mathcal{D} \subset \mathbb{R}^{n_d}$, as shown in (2.1). Given an input signal $\mathbf{u} : [0, \infty) \to \mathcal{U}$ and a disturbance signal $\mathbf{d} : [0, \infty) \to \mathcal{D}$, which return the values of the input and disturbance over time, and an initial state x, the trajectory of the system state is given by $\zeta : [0, \infty) \to \mathbb{R}^n$, as in (2.2).



(a) Dubins Car Dynamics



(b) 3D Reachability



(c) 2D Slice of Safe Set

Figure 5.2: Reachability with Dubins Car dynamics. (a) illustrates the dynamics for a Dubins car system, as described by (5.9), with a keep out set $\mathcal{K}$ that is circular in the x-y plane. (b) shows the backwards reachable tube $\mathcal{R}$ of unsafe states for this system, which is the complement of the safe set Ω. (c) illustrates a two-dimensional slice of the safe set taken when $\theta = 0$.

Safety is defined with respect to avoiding a "keep-out" set of states $\mathcal{K} \subset \mathbb{R}^n$ which represent dangerous or undesirable outcomes. If we model the input and the disturbance as attempting to drive the system away from and towards $\mathcal{K}$, respectively, then we can define the value V of this game between them as the least signed distance that will occur between $\zeta(t)$ and $\mathcal{K}$, as shown in (2.4). Under some assumptions discussed in Section 2.1.2, this value

becomes independent of time, and we have a mapping from states to values $V : \mathbb{R}^n \to \mathbb{R}$. The states with non-negative value are the ones from which the input is guaranteed to be able to avoid $\mathcal{K}$, and we denote this *safe set* by $\Omega = \{x \in \mathbb{R}^n : V(x) \geq 0\}$. Conversely, the set of states where the input cannot guarantee safety is denoted as $\mathcal{R} = \{x \in \mathbb{R}^n : V(x) < 0\}$ and is also known as the backwards reachable tube of $\mathcal{K}$. The relationship between $\mathcal{K}$, Ω, and $\mathcal{R}$ for a Dubins car system is depicted in 5.2c.

Assuming that the system is initialized within the safe set Ω, one way in which we can guarantee the continued safety of the system is by applying a minimally invasive safety controller, described in detail in Section 2.1.3. Such a controller allows for any input while $\zeta(t) \in \text{interior}(\Omega)$, but applies the optimal avoidance input, given by (2.7), on the boundary of Ω when $V(\zeta(t)) = 0$. Such a control policy grants the system as much flexibility as possible while maintaining safety guarantees.

5.1.2 Noisy Idealized Supervisor Model

We define an idealized model of the supervisor of a robotic team whose responsibility it is to ensure that no robots collide with obstacles represented by the keep-out set $\mathcal{K}$. The idealized supervisor behaves as a minimally invasive controller as described in Section 2.1.3. However, while the robotic team members' true dynamics are given by the function $f(x, u, d)$ as in (2.1), the supervisor possesses an internal model of the robots' dynamics given by $f_S(x, u, d)$, which is not necessarily equal to the true dynamics. Following the differential game characterized by (2.5), the supervisor also possesses an internal value function $V_S(\cdot)$ and safe set Ω_S which they use to evaluate the safety of each state x in the environment. We allow for the possibility that the supervisor adds some amount of margin μ to their internal safe set, such that $\Omega_S = \{x \in \mathbb{R}^n : V_S(x) \geq \mu\}$. Therefore, the idealized supervisor will always intervene when a robotic team member reaches the μ level set of $V_S(\cdot)$, rather than the zero level set of the true $V(\cdot)$. We further specify that the idealized supervisor is *conservative*: $\forall x \in \mathbb{R}^n,\ V(x) \leq 0 \implies V_S(x) \leq \mu$. This condition implies that the supervisor will never let a robot teammate leave the true safe set Ω since $\Omega_S \subseteq \Omega$. Additionally, we propose a noisy version of this idealized supervisor which does not always intervene exactly on the μ level set. A common model for noisily rational behavior is to use the Boltzmann distribution, which assigns probabilities to available action in proportion to the exponentiated values of those actions [81]. However, in this setting, applying a Boltzmann distribution to potential intervention locations x, using the values $V_S(x)$, would model the supervisor as being most likely to intervene as early and as far away from the obstacle as possible, which conflicts with our model of the supervisor as a minimally invasive safety controller. We instead select the normal distribution, which will peak at μ and distribute interventions around that point with an exponential decrease in probability based on the squared distance to μ. With this model, the noisy idealized supervisor will intervene when they observe a robot reach the $\mu + w$ level set of $V_S(\cdot)$, where w is drawn from $\mathcal{N}(0, \sigma_S^2)$ whenever a supervisor makes a safety judgement.

5.1.3 Learning Safe Sets from Supervisor Interventions

We choose to model the human supervisor of a robotic team as approximating the behavior of the idealized supervisor model presented in Section 5.1.2. That is, the human supervisor will allow the robots to perform their task however they choose, but intervene whenever they *perceive* that a robot is approaching an obstacle $\mathcal{K}$ in the state space. Given this model, we can interpret the points at which the human intervenes as corresponding to the unknown μ level set of some value function $V_H(\cdot) : \mathbb{R}^n \to \mathbb{R}$, which characterizes the human's mental safe set Ω_H. Our goal is to use observations of human interventions to derive an estimated value function $\hat{V}_H(\cdot)$ and $\hat{\mu}$ which describe the observed behavior and induce an estimated $\hat{\Omega}_H$. We approach this task by deriving a Maximum Likelihood Estimator (MLE) of the human's mental safe set. If we assume that a human supervisor always intends to intervene at the μ level set of $V_H(x)$, but their ability to precisely intervene at this level is subject to Gaussian noise, either from observation error or variability in reaction time, then we can consider the value at an intervention point x_i as being drawn from a normal distribution centered at μ (that is, $V_H(x_i) \sim \mathcal{N}(\mu, \sigma^2)$).

Given a proposed value function $\hat{V}_H(\cdot)$ and a set of intervention points $\{x_1, x_2, \cdots, x_p\}$ with corresponding values $\{\hat{V}_H(x_1), \hat{V}_H(x_2), \cdots, \hat{V}_H(x_p)\}$, we wish to estimate the most likely μ and σ^2 to explain these interventions. Gaussian distributions induce the following probability density function for a single observation $\hat{V}_H(x_j)$

$$f\left(\hat{V}_H(x_j) \mid \mu, \sigma^2\right) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{\left(\hat{V}_H(x_j) - \mu\right)^2}{2\sigma^2}\right) \tag{5.1}$$

which leads to the following probability density for a set of p independent observations

$$\begin{aligned} f\left(\hat{V}_H(x_1), \cdots, \hat{V}_H(x_p) \mid \mu, \sigma^2\right) &= \prod_{j=1}^{p} f\left(\hat{V}_H(x_j) \mid \mu, \sigma^2\right) \\ &= \left(\frac{1}{2\pi\sigma^2}\right)^{\frac{p}{2}} \exp\left(-\frac{\sum_{j=1}^{p}\left(\hat{V}_H(x_j) - \mu\right)^2}{2\sigma^2}\right). \end{aligned} \tag{5.2}$$

The likelihood of any estimated parameter values $\hat{\mu}$ and $\hat{\sigma}^2$ being correct, given the observations and the proposed value function $\hat{V}_H(\cdot)$, is expressed as

$$\mathcal{L}\left(\hat{\mu}, \hat{\sigma}^2 \mid \hat{V}_H(\cdot)\right) = f\left(\hat{V}_H(x_1), \cdots, \hat{V}_H(x_p) \mid \hat{\mu}, \hat{\sigma}^2\right). \tag{5.3}$$

It can be shown that the values of the unknown parameters μ and σ^2 that maximize the likelihood function are given by

$$\hat{\mu}^* = \frac{1}{p}\sum_{j=1}^{p} \hat{V}_H(x_j) \qquad \text{and} \qquad \hat{\sigma}^{*2} = \frac{1}{p}\sum_{j=1}^{p}\left(\hat{V}_H(x_j) - \hat{\mu}^*\right)^2, \tag{5.4}$$

which are simply the mean and variance of the set of observations.

Notice that the estimates given by (5.4) are computed with respect to a given value function $\hat{V}_H(\cdot)$. If we were to assume that the human supervisor has a perfect model of the system dynamics, then we could simply set $\hat{V}_H(\cdot)$ to equal the true $V(\cdot)$ of the system in (2.1), and $\hat{\mu}^*$ would be the maximum likelihood estimate for the level at which the supervisor will intervene. However, it is unlikely that a human supervisor's notion of the dynamics will correspond exactly to this model, and we would like to maintain the flexibility of estimating value functions that are not strictly derived from (2.1). To this end, we define the maximum likelihood of $\hat{V}_H(\cdot)$ being the $V_H(\cdot)$ that produced our observations as $\mathcal{L}^*(\hat{V}_H(\cdot)) = \max_{\hat{\mu},\hat{\sigma}^2} \mathcal{L}(\hat{\mu}, \hat{\sigma}^2 \mid \hat{V}_H(\cdot))$. The value of $\mathcal{L}^*(\hat{V}_H(\cdot))$ is obtained by substituting the estimates from (5.4) into the probability density function from (5.2). That is,

$$\mathcal{L}^* \left(\hat{V}_H(\cdot) \right) = f \left(\hat{V}_H(x_1), \cdots, \hat{V}_H(x_p) \mid \hat{\mu}^*, \hat{\sigma}^{*2} \right). \tag{5.5}$$

We seek the most likely value function to explain our observations, which will be the value function $\hat{V}^*(\cdot)$ with the greatest maximum likelihood $\mathcal{L}^*(\hat{V}^*(\cdot))$ (the maximum over maxima)

$$\hat{V}^*(\cdot) = \underset{V(\cdot)\in\mathcal{V}}{\operatorname{argmax}} \mathcal{L}^* \left(V(\cdot) \right), \tag{5.6}$$

where $\mathcal{V}$ is the set of all possible value functions.

In order to make this optimization tractable, we can restrict ourselves to a set of value functions $\{V_\theta(\cdot)\}_{\theta\in\mathbb{R}^m}$ corresponding to a family of dynamics functions $\{f_\theta(\cdot,\cdot,\cdot)\}_{\theta\in\mathbb{R}^m}$ parameterized by $\theta \in \mathbb{R}^m$, making the optimization in question

$$\hat{V}^*(\cdot) = \underset{\theta\in\mathbb{R}^m}{\operatorname{argmax}} \mathcal{L}^* \left(V_\theta(\cdot) \right). \tag{5.7}$$

In practice, we may not be able to find an expression for the gradient of $\mathcal{L}^*(V_\theta(\cdot))$ with respect to θ, since the value function is derived from the dynamics $f_\theta(\cdot,\cdot,\cdot)$ via the differential game given by (2.5). The lack of a gradient expression restricts the use of numerical methods to solve the problem as presented in (5.7). In these cases, we can compute a representative library of b value functions $\{V_i(\cdot)\}_{i=1}^b$ corresponding to a set of b representative parameter values $\{\theta_i\}_{i=1}^b$ (see Figure 5.3 for an example library). The optimization then reduces to choosing the most likely value function from among this library

$$\hat{V}^*(\cdot) = \underset{i\in\{1,\cdots,b\}}{\operatorname{argmax}} \mathcal{L}^* \left(V_i(\cdot) \right). \tag{5.8}$$

In order to ensure that the learned safe set is conservative, we can extend our MLE to a Maximum A Posteriori (MAP) estimator by incorporating our prior belief that, regardless of the safe set that the supervisor uses to generate interventions, they do not want the robots to be unsafe with respect to the true dynamics. In this case, we maintain a uniform prior

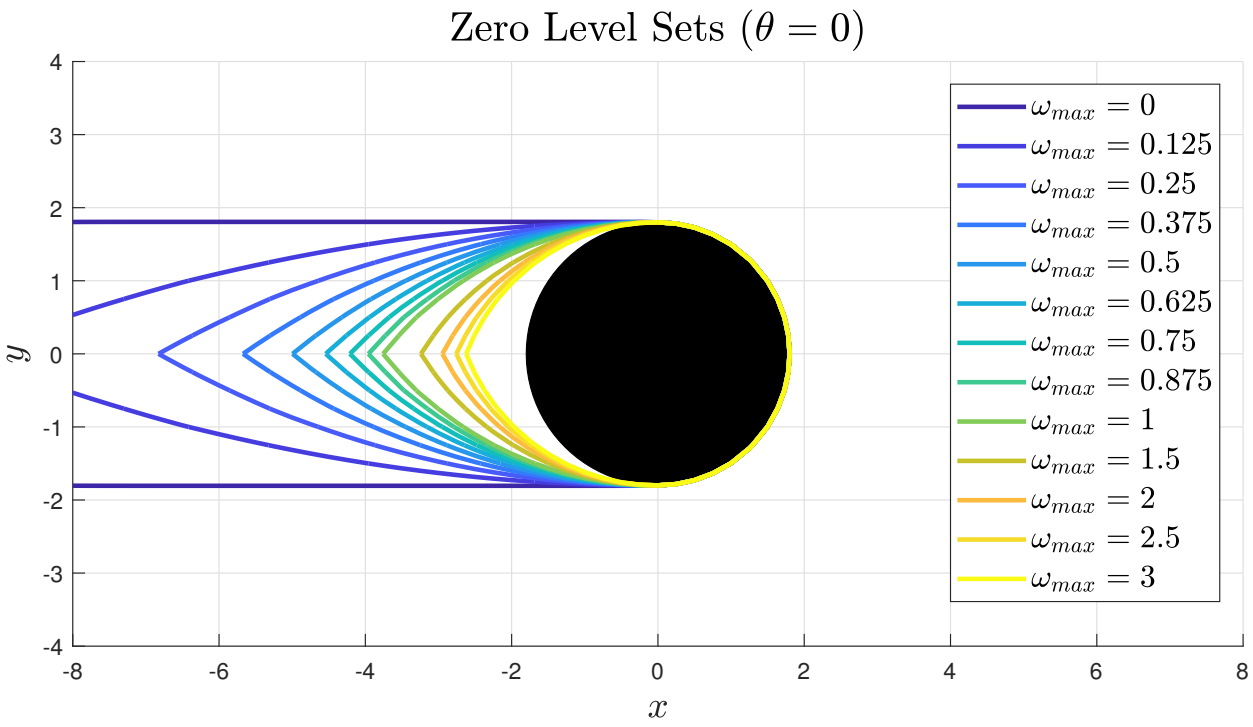


Figure 5.3: Two dimensional slices of the zero level sets of the value functions $V_i(\cdot)$ from the library used for the experiment described in Section 5.2. We used a family of Dubins car dynamics (see (5.9)) parametrized by ω_{max}. Notice that as ω_{max} decreases (the modeled control authority is decreased), the level sets extend farther away from the obstacle, indicating that a robot is expected to turn earlier to guarantee safety.

$P(\theta)$ that assigns equal probability to all $V_\theta(\cdot)$ whose zero sublevel sets are supersets of the zero sublevel set of the true $V(\cdot)$, and zero probability to all other $V_\theta(\cdot)$. In other words, we assume that the supervisor does not overestimate the agility of the robots, and in practice we can enforce this condition by choosing the library in (5.8) to only contain appropriate value functions. Moreover, regardless of the choice of $\hat{V}_H(\cdot)$, we assume that the supervisor intends to intervene before reaching the zero level set of $\hat{V}_H(\cdot)$, which always includes the boundary of $\mathcal{K}$. If we choose a prior $P(\mu)$ that assigns zero probability to all non-positive μ and uniform probability elsewhere, it can be shown that the MAP estimates are obtained by letting $\hat{\mu}^*$ equal $\max\{\hat{\mu}^*, 0\}$ and otherwise proceeding as before. Figure 5.4 provides an example of this algorithm estimating a safe set from human supervisor intervention data.

5.1.4 Team Control with Learned Safe Sets

We propose that safe sets learned according to the approach in Section 5.1.3 can be used to create effective control laws for the robotic members of human-robot teams. Recall our model of the human supervisor of a robotic team: the supervisor must rely on each robot's autonomy

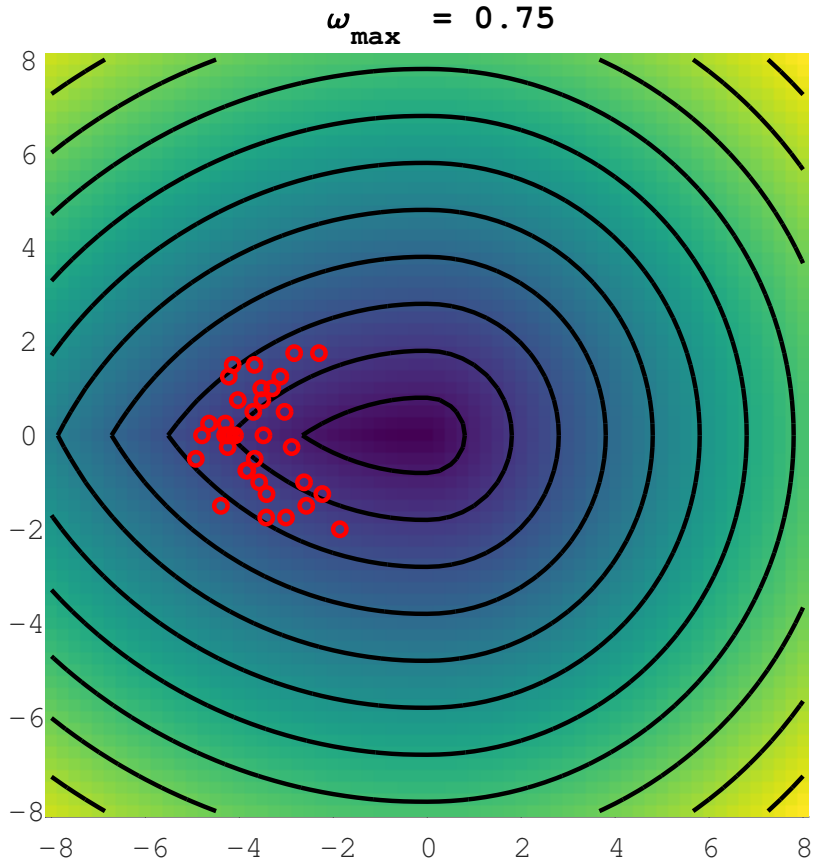


Figure 5.4: An example data set from the experiment described in Section 5.2. The red circles represent the location of supervisor interventions, and the colored background represents the learned value function $V(\cdot)$ with contour lines shown in black. In this case, the learning algorithm chose a dynamics model parametrized by $\omega_{max} = 0.75$.

to complete the majority of their tasks unassisted, but the supervisor may intervene to correct a robot's behavior when necessary (such as by avoiding an imminent collision with the keep-out set $\mathcal{K}$). We put forth that in the scenario where the human intervenes to prevent a collision, they do so because they observe that a robot has violated the boundaries of their mental safe set Ω_H.

Now, consider a team of robots navigating an unknown environment, and which are able to avoid any obstacles that they detect. One approach to safely automating this team is to have each robot behave according to a minimally invasive control law: the robots are allowed to follow trajectories generated by any planning algorithm, so long as they remain within Ω, the reachable set computed using the baseline dynamics model (2.1) with associated value function $V(\cdot)$. Whenever these robots detect an obstacle, they add it to the keep-out set

$\mathcal{K}$, thus modifying Ω and $V(\cdot)$. If a robot reaches the boundary of Ω, it applies the optimal control to avoid $\mathcal{K}$ until it has cleared the obstacle. However, it is possible that a robot does not detect an obstacle, and a human supervisor must intervene to ensure robot safety.

As stated above, the human supervisor will intervene when a robot reaches the boundary of Ω_H, not the boundary of Ω. This discrepancy leads to the possibility that the supervisor will intervene when the robot reaches some state x, even if the robot would have avoided the obstacle without intervention. These situations arise whenever $V_H(x) \leq \mu$ but $V(x) > 0$. These "false positive" interventions represent unnecessary work for the human supervisor, and we seek to eliminate them in order to improve the human's experience and the team's overall performance.

We propose using a safe set $\hat{\Omega}_H$ learned from previous observations of supervisor interventions, as outlined in Section 5.1.3, as a substitute for Ω in the robots' minimally invasive control law. By estimating the human's internal safe set, we take advantage of the following property:

Property 1. *For an idealized supervisor collaborating with a team of robots as described in Section 5.1.4, if the robots avoid detected obstacles $\mathcal{K}$ by applying an optimally safe control at the boundary of safe set Ω_S, then if the supervisor plans to intervene because they observe $\zeta_i(t) \in \mathcal{R}_S$ for robot i, the supervisor can infer that robot i has not detected an obstacle and any supervisor intervention will not be a false positive.*

Proof. The proof of this property follows constructively from the definitions of safe set, idealized supervisor, and false positive. If robot i had correctly detected an obstacle and adjusted its representation of Ω_S accordingly, then it would have applied the optimal control to remain within the supervisor's safe set. Therefore, if the supervisor is able to observe that robot i has left Ω_S, it must be the case that the robot has not detected the obstacle. False positives are defined to be supervisor interventions that occur when a robot has detected an obstacle, but the supervisor still intervenes. In this case, the supervisor can correctly infer that robot i has not detected an obstacle, so any intervention at this point cannot be a false positive. ■

For an idealized supervisor, as $\hat{\Omega}_S$ becomes an arbitrarily good approximation of Ω_S, the number of false positive interventions will approach zero. For a *noisy* idealized supervisor, the supervisor will intervene whenever $V_S(x) + w \leq \mu$ where $w \sim \mathcal{N}(0, \sigma_S^2)$. This noise will continue to produce false positives, even with a perfect fit $\hat{\Omega}_S = \Omega_S$, if the robots apply the optimally safe control at the μ level set of Ω_S. Instead, the level set α where the optimally safe control is applied can be raised arbitrarily high to drive the false positive rate to zero. For example, $\alpha = \mu + 2\sigma_S$ is sufficient to avoid over 97% of intervention states used for learning, in expectation. We test the efficacy of our approach through the human-subjects experiment described in Section 5.2.

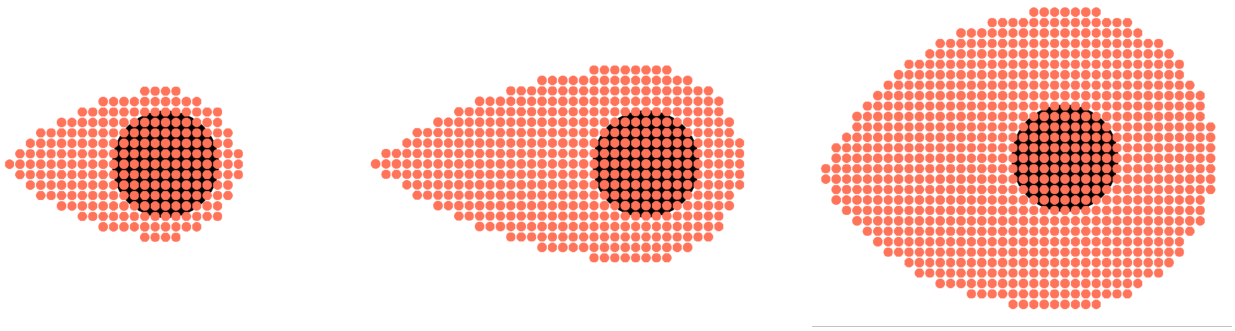


Figure 5.5: Safe sets tested in our experiment (illustrated by their complementary reachable set): (left) Standard safe set (calculated from true dynamics and obstacle size), (middle) example Learned safe set (calculated from fitted supervisory perception of dynamics and obstacle size), (right) Conservative safe set (calculated from true dynamics and inflated obstacle size)

5.2 Experimental Design for User Validation

Our goal in understanding and modeling the supervisor's conception of safety is to improve team performance by decreasing cognitive overload. Although we have based our human modeling on the cognitive science literature, we do not intend to verify humans' exact cognitive processes. Instead, we aim to apply our inspiration from cognitive science toward building better human-robot teams. To this end, our hypotheses are:

H1. *Representing supervisor behavior as cognitive keep-out sets allows intervention signals to be distilled into an actionable rule which will decrease supervisory false positives and cognitive strain, thereby increasing team performance and trust.*

H2. *Fitting danger-avoidance behavior to a supervisor's beliefs is preferable to generic conservative behavior.*

In our experiment, we gather supervisor intervention data, fit our model to the data, and then run a human-robot teaming task that assesses performance.

5.2.1 Procedure

Our experiment applies the idealized supervisor theory and learning algorithm to supervising simulated robots. The robots moved according to the Dubins car model:

$$\begin{aligned} \dot{x} &= v\cos(\theta), \\ \dot{y} &= v\sin(\theta), \\ \dot{\theta} &= u, \\ v = 3,\ u \in [-\omega_{max}, \omega_{max}],\ \omega_{max} &= 1, \end{aligned} \tag{5.9}$$

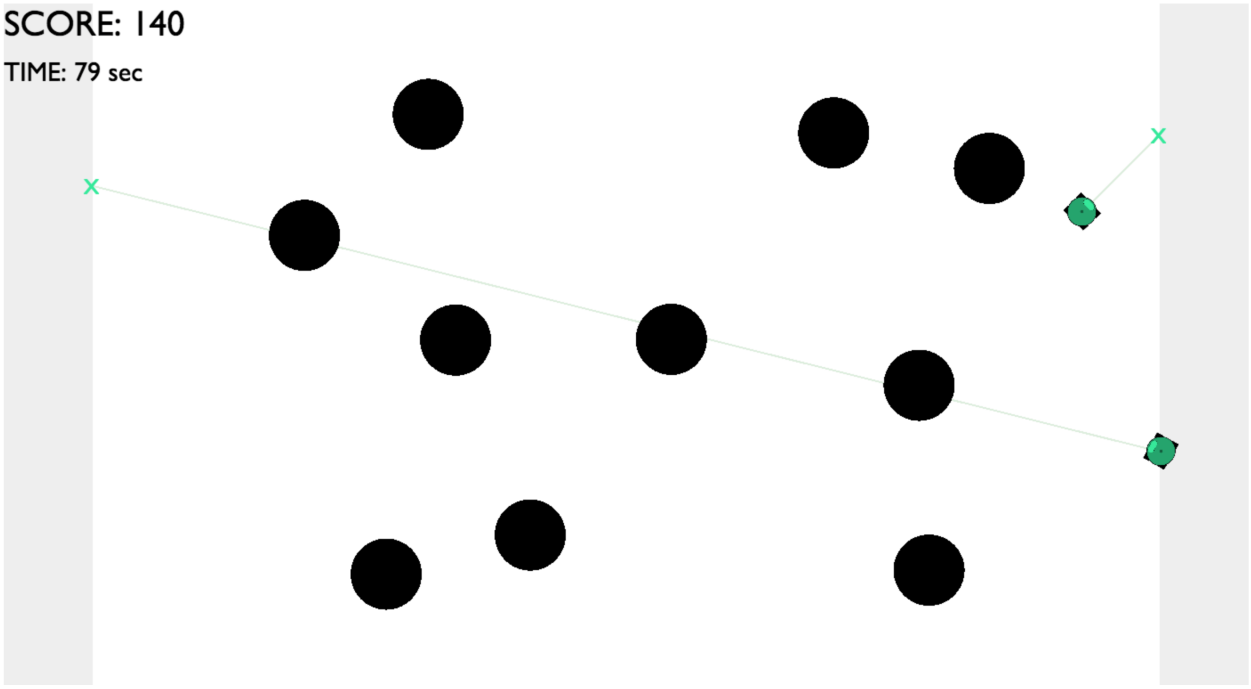


Figure 5.6: Screenshot of the task from Phase III of the experiment. Robotic vehicles make trips back and forth across the screen, detecting and avoiding each obstacle with 80% probability. The human supervisor must remove an obstacle in the event that it is undetected, but must infer this information from the robots' motion.

which is illustrated in Figure 5.2a.

The experiment is divided into three phases. In Phase I, the subject is given an opportunity to familiarize themselves with the robotic system's dynamics. The user is allowed to directly apply the full range of controls through the computer keyboard for one minute. After ensuring the user has some experience from which to build an internal dynamics model, we then assess their emergent conception of safety. In Phase II, supervisory data is extracted from the subject by showing them scenes where the robot is driving towards an obstacle, and the supervisor decides where to intervene to avoid a crash. This intervention data is then fed into our algorithm (described in Section 5.1.3) that extracts the best fitting safe set. Our estimator used a library of candidate dynamics functions parameterized by values of ω_{max} between 0 and 3, as shown in Figure 5.3. In this experiment, we enforced conservativeness by excluding subjects whose Learned sets were not supersets of the Standard safe set, rather than enforcing a prior directly on ω_{max}. The Learned safe set is assessed in Phase III against two fixed safe sets (see Figure 5.5) pre-calculated from the true dynamic equations.

These safe sets were calculated using Hamilton-Jacobi reachability as described in Section 2.1 using the Level Set Toolbox [82] for MATLAB. During this final phase, the subject sequentially supervises homogeneous teams of robots, each team avoiding obstacles based on one of the three assessed safe sets. Ten randomly placed obstacles are strewn about

the screen impeding the robots' autonomous trips back and forth across the screen (see Figure 5.6). Although robots will detect and avoid an obstacle in 80% of their interactions with it, there is a 20% chance that the robot will not detect an obstacle as it approaches. The subject is charged with catching these random failures and removing an obstacle before the robot crashes. Crashing is disincentivized by decrementing an on-screen "score" counter. Removing an obstacle costs only half of what a crash costs the player. This system encourages saving the robot but not guessing wildly. Moreover, simply clearing out all obstacles is not a viable strategy because every obstacle removed generates a new obstacle elsewhere. This score mechanism was also used to make the participant invested in team success by awarding points every time a robot completes a trip across the screen.

5.2.2 Independent Variables

To assess our hypotheses, we manipulate the safe set used between team supervision trials. We exposed the human subject to three teams, each driving using one of three safe sets. The Learned set is derived from Phase II supervisor intervention observations as described in Section 5.1, using $\alpha = \mu$. The two baseline kernels are calculated using Hamilton-Jacobi-Isaacs reachability on the true dynamic equations. The Standard set is calculated using the true obstacle size. The Conservative set adds a buffer that doubles the effective size of the obstacle, inducing trajectories that give obstacles a wide berth.

5.2.3 Dependent Measures

Objective Measures

The team was tasked with making trips across the screen to reach randomized goals. The robots' task was to travel across the screen, safely dodging obstacles along the way, while the human was tasked with supervising as a failsafe to remove an obstacle if the robots should fail to observe and avoid it.

Team performance was quantified using three objective metrics: number of trips completed, number of supervisory interventions, and the number of obstacle collisions. These metrics were presented to the subject as an aggregated, arcade-style score. To incentivize participants to only intervene when necessary, obstacle-removal interventions reduced the score, but only by half as much as an obstacle collision.

The number of interventions taken by the supervisor can also serve as a proxy measurement to quantify the amount of cognitive strain they experience while working with the robotic team. Of particular note are the number of interventions that were not actually required, as the supervisor incorrectly judged that a robot had not detected an obstacle. These false positives needlessly drain supervisor attention and indicate a lack of trust in the system. We aim to increase the human's trust in the system, which we quantify by a decrease in these false positives.

Subjective Measures

After each round of pairwise comparison (completing the task with two different robotic teams), we presented the subject with a questionnaire to gauge how the choice of safe set impacted their experience. These questionnaires contained statements about each team that subjects would respond to using a 7-point Likert scale (1 - Strongly Disagree, 7 - Strongly Agree). These statements were designed to measure Trust, Perceived Performance, Interpretability, Confidence, Team Fluency, and overall Preference between the teams in the comparison.

5.2.4 Subject Allocation

The subject population consisted of 6 male, 5 female, and 1 non-binary participants between the ages of 18-29. We used a within-subjects design where each subject was asked to complete all three possible pairwise comparisons of our three treatments (the safe sets used). We used a balanced Latin Square design for the order of comparisons, with no treatment being first in a pair twice. Furthermore, we generated six randomized versions of the task so that subjects were presented with a different version of the task for each trial across the three pairwise comparisons. To avoid coupling the treatment results to a particular version of the task, each treatment was paired with each task version an equal number of times across our subject population.

5.3 Analysis and Discussion

5.3.1 H1: False Positive Reduction over Standard

Our first hypothesis is that a Learned safe set that reflects the supervisor's intervention behavior would decrease the number of false positives compared to the Standard safe set. To test this, we performed a one-way repeated measures ANOVA on the number of supervisory false positives from Phase III of the experiment with safe set as the manipulated factor. A false positive was any supervisor intervention where the removed obstacle was actually detected by all nearby robots, which would have avoided it successfully. The robot team's safe set had a significant effect on the number of supervisory false positives ($F(2, 20) = 8.72$, $p < 0.01$). An all-pairs post-hoc Tukey method found that the Learned safe set significantly decreased ($p = 0.0328 < 0.05$) false positives over the Standard safe set, but there was no significant difference between the Learned safe set and the Conservative safe set (which also significantly decreased false positives over the Standard safe set, with $p < 0.01$). These results support our main hypothesis that ***representing supervisor behavior as cognitive keep-out sets allows intervention signals to be distilled into an actionable rule which will decrease supervisory false positives***.

The second half of that hypothesis, that ***decreasing supervisory false positives will increase trust and team performance*** was not shown conclusively from our data.

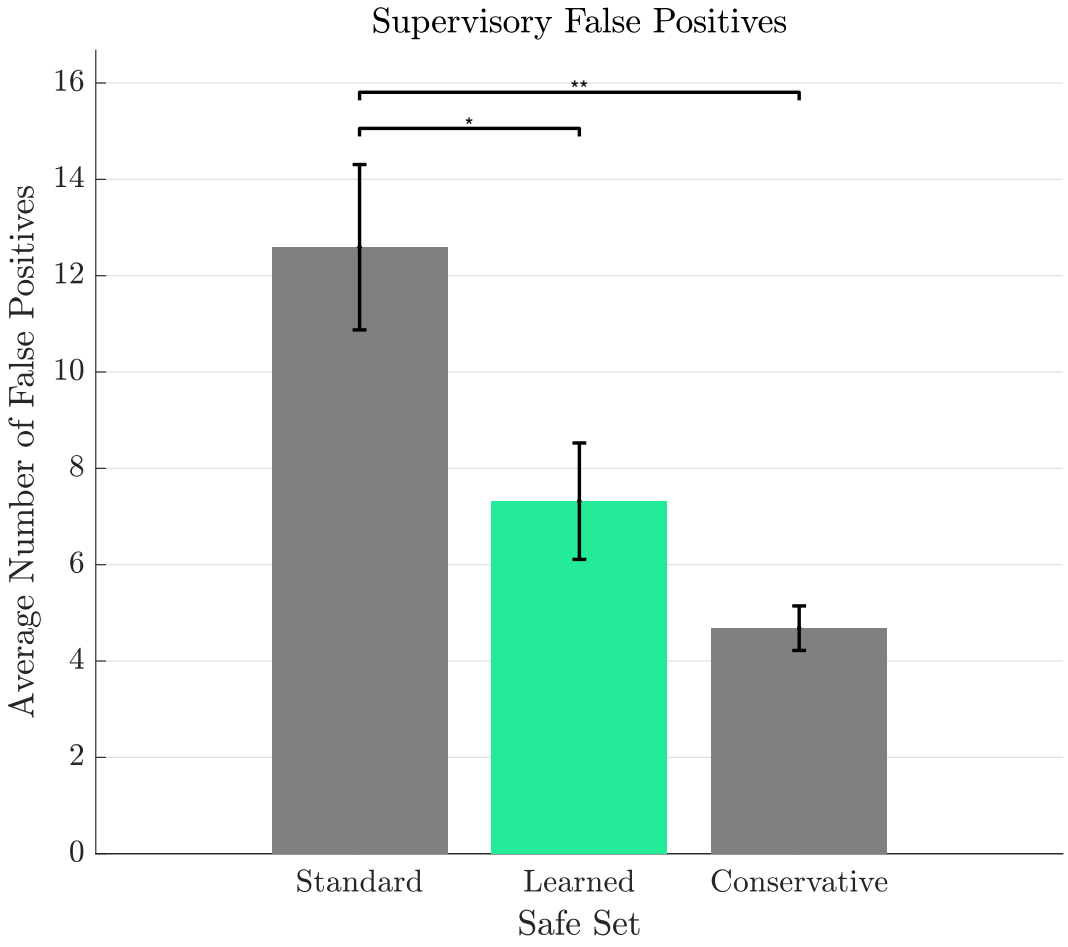


Figure 5.7: Average number of false positives per trial plotted against the three safe set types. There were significant differences between Standard and Learned ($p < .05$) and between Standard and Conservative ($p < .01$). There was no significant difference between Learned and Conservative.

We performed a one-way, repeated measures ANOVA on the pairwise comparison surveys between the teams using the Learned and the Standard safe sets. Measures of trust showed no significant improvement ($F(1, 9) = 1.86$, $p = 0.21$).

5.3.2 H2: Preference over Conservative

For 9 of 11 participants, the Learned safe set had shorter avoidance arcs than the Conservative set. We hypothesized that this greater efficiency would make the tailored conservativeness of the Learned set preferable to the baseline Conservative safe set. However, a t-test showed that the survey responses for preference were statistically indistinguishable

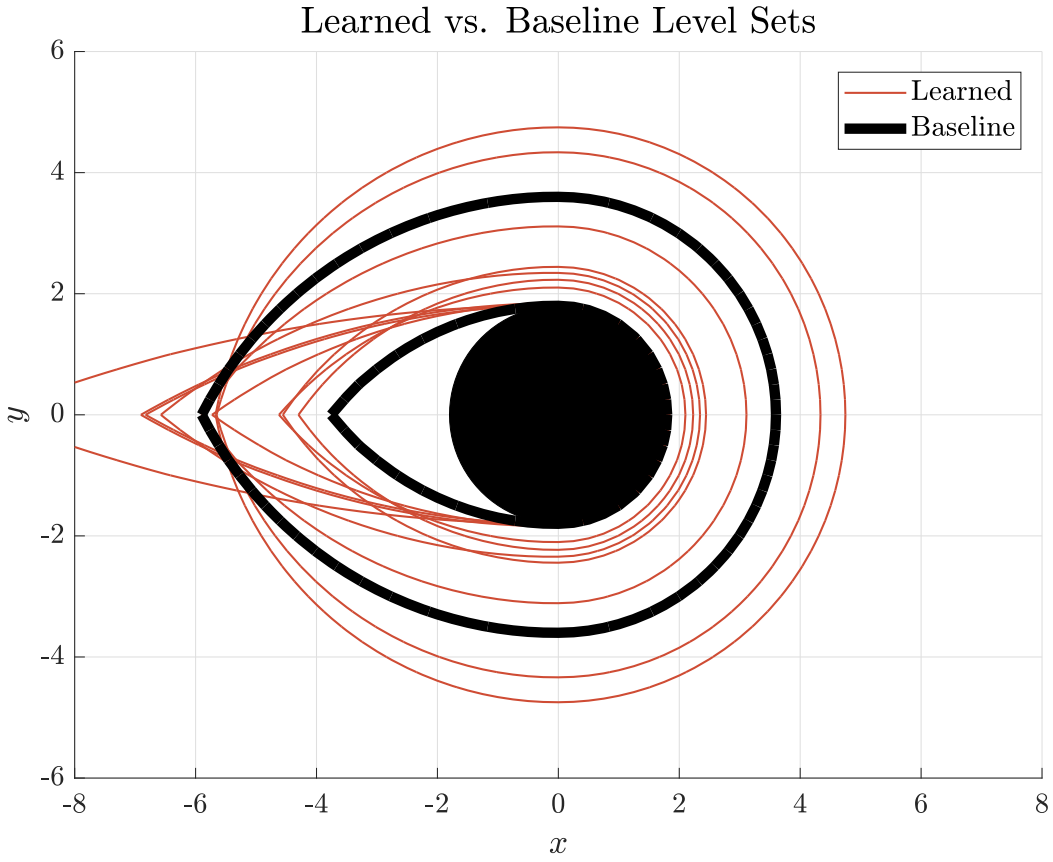


Figure 5.8: Regressed safe sets (viewed on the $\theta = 0$ slice) from supervisor intervention data overlaid on baselines. Three users' safe sets clustered to arcing like the Standard safe set. Three others clustered to arcing like the Conservative safe set. The final five safe sets exhibit a distinct behavior that reflects supervisors' preference for gradual, pre-emptive arcs.

($p = 0.8$) from a neutral score: an inconclusive result for Hypothesis 2. We believe that this result stems from users judging preference more on intelligibility, the ease of avoiding false positives, than on efficiency, the shortness of paths. As discussed in Section 5.3.1, both the Learned and Conservative safe sets led to significant false positive reductions over the Standard set.

This indistinguishability is further compounded since a preference for intelligibility seems to be expressed by some subjects in their Phase II intervention data, resulting in their Learned safe sets having similar arcs as the Conservative safe set (see Figure 5.8). Future work could investigate this efficiency-intelligibility trade-off further by using a conservative baseline that is distinguishably more conservative than user safe sets and by making efficiency more central to the team task.

5.3.3 Model Validity

The statistically significant decreases in false positives observed in Phase III agree with the decreases predicted by the supervisor model based on intervention data from Phase II. Our model posits that interventions occur at states noisily distributed about a safe set boundary. Therefore, it predicts that the empirical distribution of Phase II intervention states contained within a proposed safe set (see Figure 5.9) will mirror the proportion of false positive interventions observed in Phase III: if states are deemed safe by the controller, they will not be avoided, even when the noisy supervisor would judge them to be unsafe. Since the Learned safe set controller intervenes at the $\hat{\mu}^*$ level set (see Section 5.1.3), exactly half the intervention states will be contained within the Learned safe set in expectation. The model's predictions are compared against observed false positives in Table 5.1.

	Interventions in Safe Set	Predicted F.P. vs Std.	Average F.P.	Observed F.P. vs Std.
Standard	397 / 440	100%	12.54	100%
Learned	220 / 440	55.4%	7.31	58.3%
Conservative	115 / 440	29%	4.68	37.3%

Table 5.1: Predicted and observed false positives. Left: Predicted false positives from Phase II data. Right: Observed false positives in Phase III.

5.4 Conclusion

Automation with human supervisors relies on leveraging the human supervisor's cognitive resources for success. Respecting these resources is essential for creating well performing human-robot teams. It is especially important to avoid overtaxing the human as automated teams continue to scale up, and a single human worker both accomplishes more and bears more cognitive load than ever. To alleviate this burden, we can decrease the number of issues that command the supervisor's attention by reducing false positives. By modeling which system states command supervisory attention, we can program autonomous systems to avoid those states when they do not require attention. To capture this information, we combine the concept of mental simulation from cognitive science with formal safety analysis from reachability theory to propose the noisy idealized supervisor model. We employ the noisy idealized supervisor as the generative model in a learning algorithm to predict supervisor safety judgements, and we present a safety controller for robotic agents that respects the supervisor's perception of safety. This safety controller is guaranteed to reduce false positives for idealized supervisors. Furthermore, for actual supervisors, our human-robot teaming user study demonstrated a significant reduction in false positives when using our approach compared to the standard baseline.

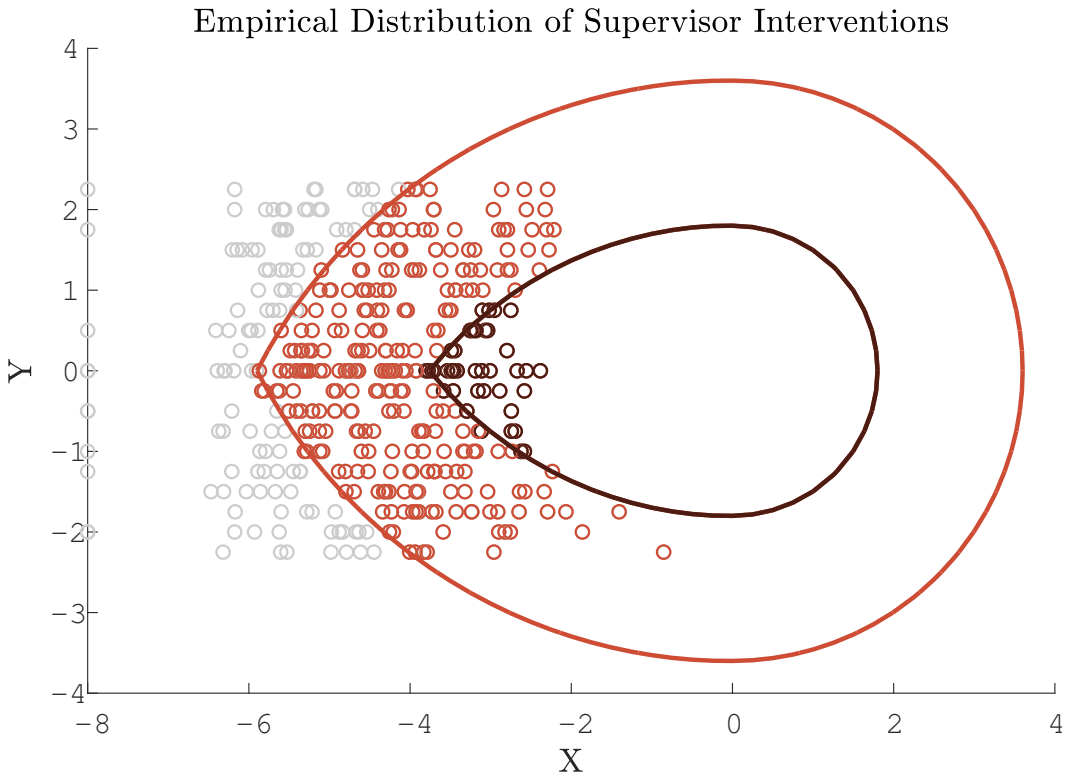


Figure 5.9: Empirical distribution of intervention states observed during data collection (Phase II of the experiment). The interventions within the Conservative reachable set are colored in red, leaving 115 interventions in the corresponding safe set. Similarly, the interventions within the Standard reachable set are colored darker, leaving 397 interventions in the corresponding safe set. Intervention states not contained within a reachable set would have generated a false positive during the human-robot teaming task.

Our results show that it is possible to reduce false positives, and thus cognitive load, by aligning robot behavior with humans' expectations. Our approach is applicable whenever reachability theory can tractably analyze a dynamical system that will be subject to human safety judgements. Future work will explore the impact of this framework on application domains from air traffic management to self-driving vehicles.

Chapter 6

Maximum Likelihood Constraint Inference

Advances in mechanical design and artificial intelligence continue to expand the horizons of robotic applications. In these new domains, it can be difficult to design a specific robot behavior by hand. Even manually specifying a task for a reinforcement-learning-enabled agent is notoriously difficult [47], [48]. Inverse Reinforcement Learning (IRL) techniques, discussed in Section 2.3, can help alleviate this burden by automatically identifying the objectives driving certain behavior. Since first being introduced as Inverse Optimal Control (IOC) by Kalman [46], much of the work on IRL has focused on learning environmental rewards to represent the task of interest [1]–[3], [5]. While these types of IRL algorithms have proven useful in a variety of situations [13], [53], [62], [83], their basis in assuming that reward functions fully represent task specifications makes them ill suited to problem domains with hard constraints or non-Markovian objectives.

Recent work has attempted to address these pitfalls by using demonstrations to learn a rich class of possible specifications that can represent a task [84]. Others have focused specifically on learning constraints, that is, behaviors that are expressly forbidden or infeasible [14], [85]–[88]. Such constraints arise in safety-critical systems, where requirements such as an autonomous vehicle avoiding collisions with pedestrians are more naturally expressed as hard constraints than as soft reward penalties. It is towards the problem of inferring such constraints that we turn our attention. In this work, we present a novel method for inferring constraints, drawing primarily from the Maximum Entropy approach to IRL [5]. We use this framework to reason about the likelihood of observing a set of demonstrations given a nominal task description, as well as about their likelihood if we imposed additional constraints on the task. This knowledge allows us to select a constraint, or set of constraints, which maximizes the demonstrations' likelihood and best explains the differences between expected and demonstrated behavior. Our method improves on prior work by being able

The work in this chapter first appeared in "Maximum Likelihood Constraint Inference for Inverse Reinforcement Learning" [15].

to simultaneously consider constraints on states, actions and features in a Markov Decision Process (MDP) to provide a principled ranking of all options according to their effect on demonstration likelihood.

6.1 Related Work

6.1.1 Inverse Reinforcement Learning

As presented by Kalman [46], the objective of IOC/IRL is to use knowledge about a dynamical system and a control law governing that system in order to infer which function this controller was designed to optimize. More recent work on IRL has focused specifically on modeling systems as MDPs and using expert demonstrations to infer the reward function which these demonstrations attempt to optimize, as in (2.16) [1]–[3], [5], [50]–[52], [89]. While many different formulations of IRL have been developed, they almost universally focus specifically on learning *Markovian* rewards, which depend only on the current state and/or action.

One of the main difficulties of using IRL to learn these rewards is the fact that potentially infinite reward functions can make demonstrated behavior appear optimal. Resolving this ambiguity has received a lot of attention in IRL research, and we discuss some specific examples in Section 2.3.2. In particular, Ziebart, Maas, Bagnell, *et al.* [5] resolve the ambiguity by allowing demonstrations to be *noisily* optimal with respect to rewards and employing the principle of maximum entropy [49]. Their resulting Maximum Entropy IRL (MaxEnt) technique, which we discuss in detail in Section 2.3.3, induces a probability distribution over possible trajectories, according to (2.23), in which the likelihood of an agent producing a trajectory increases exponentially with the reward earned along that trajectory. This approach allows the authors to learn a reward function that will match the expected feature counts from the demonstrations without injecting any additional preferences between possible trajectories. This induced probability distribution over trajectories has been broadly adopted by more recent IRL techniques [50]–[52], and it is central to our efforts in identifying the most likely behavior-modifying constraints.

6.1.2 Beyond Reward Functions

While Markovian rewards do often provide a succinct and expressive way to specify the objectives of a task, they cannot capture all possible task specifications. Vazquez-Chanlatte, Jha, Tiwari, *et al.* [84] highlight the utility of non-Markovian Boolean specifications which can describe complex objectives (e.g. do this before that) and compose in an intuitive way (e.g. avoid obstacles *and* reach the goal). The authors of that work draw inspiration from the MaxEnt framework to develop their technique for using demonstrations to calculate the posterior probability that an agent is attempting to satisfy a Boolean specification.

A subset of these types of specifications that is of particular interest to us is the specification of constraints, which are states, actions, or features of the environment that must be avoided. Chou, Berenson, and Ozay [88] explore how to infer trajectory feature constraints given a nominal model of the environment (lacking the full set of constraints) and a set of demonstrated trajectories. The core of their approach is to sample from the set of trajectories which have better performance than the demonstrated trajectories. They then infer that the set of possible constraints is the subset of the feature space that contains the higher-reward sampled trajectories, but not the demonstrated trajectories. Intuitively, they reason that if the demonstrator could have passed through those features to earn a higher reward, but did not, then there must have been a previously unknown constraint preventing that behavior. However, while their approach does allow for a cost function to rank elements from the set of possible constraints, the authors do not offer a mechanism for determining what cost function will best order these constraints.

Our approach to constraint inference from demonstrations addresses this open question by providing a principled ranking of the likelihood of constraints. We adapt the MaxEnt framework to allow us to reason about how adding a constraint will affect the likelihood of demonstrated behaviors, and we can then select the constraints which maximize this likelihood. We consider feature-space constraints as in [88], and we explicitly augment the feature space with state- and action-specific features to directly compare the impacts of state-, action-, and feature-based constraints on demonstration likelihood.

6.2 Maximum Likelihood Constraint Inference

6.2.1 Problem Formulation

Following the formulation presented by Ziebart, Maas, Bagnell, *et al.* [5], we base our work in the setting of a (finite-state) Markov Decision Process (MDP). We define an MDP $\mathcal{M}$ as a tuple $(S, \{A_s\}, \{P_{s,a}\}, D_0, \phi, R)$ as in Section 2.3.1, with the specific definitions of those elements given here as:

- S is a finite set of discrete states;
- $\{A_s\}$ is a set of the sets of actions available to be taken for each state s, such that $A_s \subseteq A$, where A is a finite set of discrete actions;
- $\{P_{s,a}\}$ is a set of state transition probability distributions such that $P_{s,a}(s') = P(s'|s,a)$ is the probability of transitioning to state s' after taking action a from state s;
- $D_0 : S \to [0,1]$ is an initial state distribution;
- $\phi : S \times A \to \mathbb{R}^k_+$ is a mapping to a k-dimensional space of non-negative features; and
- $R : S \times A \to \mathbb{R}$ is a reward function.

A trajectory ξ through this MDP is a sequence of states s_t and actions a_t such that $s_0 \sim D_0$ and state $s_{i+1} \sim P_{s_i,a_i}$. Actions are chosen by an agent navigating the MDP according to a, potentially time-varying, policy π such that $\pi(\cdot|s,t)$ is a probability distribution over actions in A_s. We denote a finite-time trajectory of length $T+1$ by $\xi = \{\mathbf{s}_{0:T}, \mathbf{a}_{0:T}\}$.

At every time step t, a trajectory will accumulate features equal to $\phi(s_t, a_t)$. We use the notation $\phi_i(\cdot,\cdot)$ to refer to the i-th element of the feature map, and we use the label ϕ_i to denote the i-th feature itself. We also introduce an augmented indicator feature mapping $\widetilde{\phi}^{\mathbb{1}} : S \times A \to \{0,1\}^{n_\phi}$, where $n_\phi = k + |S| + |A|$. This augmented feature map uses binary variables to indicate the presence of a feature and expands the feature space by adding binary features to track occurrences of each state and action, such that

$$\widetilde{\phi}^{\mathbb{1}}_{\phi_i}(s,a) = \begin{cases} 1 & \text{if } \phi_i(s,a) > 0 \\ 0 & \text{otherwise} \end{cases}, \quad \widetilde{\phi}^{\mathbb{1}}_{s_i}(s,a) = \begin{cases} 1 & \text{if } s = s_i \\ 0 & \text{otherwise} \end{cases}, \quad \widetilde{\phi}^{\mathbb{1}}_{a_i}(s,a) = \begin{cases} 1 & \text{if } a = a_i \\ 0 & \text{otherwise} \end{cases}. \tag{6.1}$$

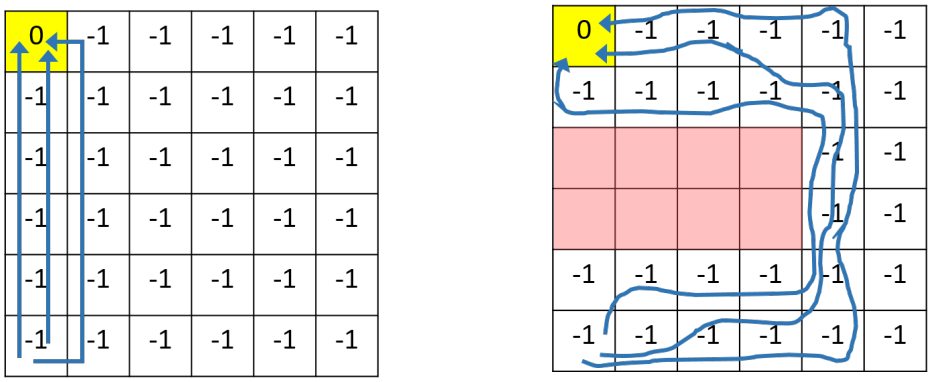
Typically, agents are modeled as trying to maximize, or approximately maximize, the total reward earned for a trajectory ξ, given by $R(\xi) = \sum_{t=0}^{T} \gamma^t R(s_t, a_t)$, where $\gamma \in (0,1]$ is a discount factor. Therefore, an agent's policy π is closely tied to the form of the MDP's reward function.

Conventional IRL focuses on inferring a reward function that explains an agent's policy, revealed through the behavior observed in a set of demonstrated trajectories $\mathcal{D}$. However, our method for constraint inference poses a different challenge: given an MDP $\mathcal{M}$, including a reward function, and a set of demonstrations $\mathcal{D}$, find the most likely set of constraints C^* that could modify $\mathcal{M}$ to explain these demonstrations. We define our notion of constraints in the following section.

6.2.2 Constraints for MDPs

Constraints are those behaviors that are not disallowed explicitly by the structure of the MDP, but which would be infeasible or prohibited for the underlying system being modeled by the MDP. This sort of discrepancy can occur when a generic or simplified MDP is designed without exact knowledge of specific constraints for the modeled system. For instance, for a generic MDP modeling the behavior of cars, we might want to include states for speeds up to 500km/h and actions for accelerations up to 12m/s^2. However, for a specific car on a specific roadway, the set of states where the vehicle travels above 100km/h may be prohibited because of a speed limit, and the set of actions where the vehicle accelerates above 4m/s^2 may be infeasible because of the physical limitations of the vehicle's engine. Therefore, any MDP trajectory of this specific car system would not contain a state-action pair which violates these legal and physical limits. Figure 6.1 shows an example of constraints driving behavior.

We define a constraint set $C_i \subseteq S \times A$ as a set of state-action pairs that violate some specification of the modeled system. We consider three general classes of constraints: state constraints, action constraints, and feature constraints.



(a) Nominal MDP (b) Constrained MDP

Figure 6.1: Illustration of trajectories likely to be produced by noisily optimal agents navigating an MDP. (a) Expected behavior on a generic, nominal MDP. (b) Demonstrated behavior from a specific, constrained MDP. Numbers represent state-based rewards, and the red-shaded tiles represent state constraints.

- A state constraint set C_{s_i} includes all state-action pairs such that the state component is s_i. That is, $C_{s_i} = \{(s, a) \mid s = s_i\}$.
- An action constraint set C_{a_i} includes all state-action pairs such that the action component is a_i. That is, $C_{a_i} = \{(s, a) \mid a = a_i\}$.
- A feature constraint set C_{ϕ_i} includes all state-action pairs that produce a non-zero value for feature ϕ_i. That is, $C_{\phi_i} = \{(s, a) \mid \phi_i(s, a) > 0\}$.

If we augment the set of features as described in (6.1), it is straightforward to see that state and action constraints become special cases of feature constraints, with constraint sets given by $C_i = \{(s, a) \mid \widetilde{\phi}_i^{\mathbb{1}}(s, a) = 1\}$. It is also evident that we can obtain compound constraints, respecting two or more conditions, by taking the union of constraint sets C_i to obtain $C = \bigcup_i C_i$.

Adding Constraints to an MDP

We need to be able to reason about how adding a constraint to an MDP will influence the behavior of agents navigating that environment. If we impose a constraint on an MDP, then none of the state-action pairs in that constraint set may appear in a trajectory of the constrained MDP. To enforce this condition, we must restrict the actions available in each state so that it is not possible for an agent to produce one of the constrained state-action pairs. For a given constraint C, we can replace the set of available actions A_s in every state s with an alternative set A_s^C given by

$$A_s^C = A_s \setminus \{a \in A_s \mid (s, a) \in C\}. \tag{6.2}$$

Performing such substitutions for an MDP $\mathcal{M}$ will lead to a modified MDP $\mathcal{M}^C$ such that $\mathcal{M}^C = (S, \{A_s^C\}, \{P_{s,a}\}, D_0, \phi, R)$.

The question then arises as to the how we should treat states with empty action sets $A_s^C = \emptyset$. Since an agent arriving in such an *empty state* would have no valid action to select, any trajectory visiting an empty state must be deemed invalid. Indeed, such empty action sets will be produced for any state s_i such that $C_{s_i} \subseteq C$.

For MDPs with deterministic transitions, agents know precisely which state they will arrive in following a certain action. Therefore, any agent respecting constraints will not take an action that leads to an empty state, since doing so will lead to constraint violations. If we consider the set of empty states S_{empty}, then for the purposes of reasoning about an agent's behavior, we can impose an additional constraint set

$$C_{\text{empty}} = \{(s, a) \mid \exists\, s_{\text{empty}} \in S_{\text{empty}} : P_{s,a}(s_{\text{empty}}) = 1\}. \tag{6.3}$$

In this work, we will always implicitly add this constraint set, such that $\mathcal{M}^C$ will be equivalent to $\mathcal{M}^{C \cup C_{\text{empty}}}$, and we recursively add these constraints until reaching a fixed point.

For MDPs with stochastic transitions, the semantics of an empty state are less obvious and could lend themselves to multiple interpretations depending on the nature of the system being modeled. In our context, we use constraints to describe how observed behaviors from *demonstrations* differ from possible behaviors allowed by the nominal MDP structure. We therefore assume that any demonstrations provided are, by the fact that they were selected to be provided, consistent with the system's constraints, including avoiding empty states. This assumption implies that any stochastic state transitions that would have led to an empty state will not be observed in trajectories from the demonstration set. The omission of these transitions means that, for a given (s, a), if $P_{s,a}(S_{\text{empty}}) = p$, then a proportion p of these (s, a) pairs which occur as an agent navigates the environment will be excluded from demonstrations. Therefore, as we modify the MDP to reason about demonstrated behavior, we need updated transition probabilities which eliminate the probability mass of transitioning to empty states, an event which will never be observed in a demonstration. Such modified probabilities can be given as

$$P_{s,a}^C(s') = \begin{cases} 0 & \text{if } s' \in S_{\text{empty}} \\ \frac{Ps,a(s')}{1 - Ps,a(S_{\text{empty}})} & \text{otherwise} \end{cases}. \tag{6.4}$$

We must also capture the change to observed state-action pair frequencies by understanding that any observed policy π^C will be related to an agent's actual policy π according to

$$\pi^C(a|s, t) = \frac{\pi(a|s, t)(1 - P_{s,a}(S_{\text{empty}}))}{\sum_{a' \in A_s^C} \pi(a'|s, t)(1 - P_{s,a}(S_{\text{empty}}))}. \tag{6.5}$$

It is important to note that the modifications presented in (6.4) and (6.5) for stochastic MDPs are not meant to directly reflect the reality of the underlying system (we wouldn't

expect the actual transition dynamics to change, for instance), but to reflect the *apparent* behavior that we would expect to observe in the subset of trajectories that would be selected as demonstrations. We further note that applying these modifications to deterministic MDPs will result in the same expected behavior as augmenting the constraint set with C_{empty}.

Nominal MDPs

The nominal MDPs to which we will add constraints fall into two broad categories, which we denote as *generic* and *baseline*. The car MDP described at the beginning of Section 6.2.2 is an example of a generic nominal model: its state and action spaces are broad enough to encompass a wide range of car models, and we can use this nominal model to infer constraints that specialize the MDP to a specific car and task. For a generic nominal MDP, the reward function may also come from a generic, simplified task, such as "minimize time to the goal" or "minimize energy usage."

A baseline nominal MDP is a snapshot of a system from a point in time where it was well characterized. With a baseline nominal MDP, the constraints that we infer will represent changes to the system with respect to this baseline. In this case, the nominal reward function can be learned using existing IRL techniques with demonstrated behavior from the baseline model. We take this approach in our human obstacle avoidance example in Section 6.3.2: we use demonstrations of humans walking through the empty space to learn a nominal reward, then we can detect the presence of a new obstacle in the space from subsequent demonstrations.

6.2.3 Demonstration Likelihood Maximization

Our goal is to find the constraints C^* which are most likely to have been added to a nominal MDP $\mathcal{M}$, given a set of demonstrations $\mathcal{D}$ from an agent navigating the constrained MDP. Let us define $P_{\mathcal{M}}$ to denote probabilities given that we are considering MDP $\mathcal{M}$. Our problem then becomes to select the constraints that maximize $P_{\mathcal{M}}(C \mid \mathcal{D})$. If we assume a uniform prior over possible constraints, then we know from Bayes' Rule that $P_{\mathcal{M}}(C \mid \mathcal{D}) \propto P_{\mathcal{M}}(\mathcal{D} \mid C)$. Therefore, in order to find the constraints that maximize $P_{\mathcal{M}}(C \mid \mathcal{D})$, we can solve the equivalent problem of finding which constraints maximize the likelihood of the given demonstrations. In this section, we present our approach to solving maximum likelihood constraint inference via solving demonstration likelihood maximization.

Under the maximum entropy model presented in [5], the probability of a certain finite-length trajectory ξ being executed by an agent traversing a deterministic MDP $\mathcal{M}$ is exponentially proportional to the reward earned by that trajectory.

$$P_{\mathcal{M}}(\xi) = \frac{1}{Z} e^{\beta R(\xi)} \mathbb{1}^{\mathcal{M}}(\xi), \tag{6.6}$$

where Z is the *partition function*, $\mathbb{1}^{\mathcal{M}}(\xi)$ indicates if the trajectory is feasible for this MDP, and $\beta \in [0, \infty)$ is a parameter describing how closely an agent adheres to the task of optimizing the reward function (as $\beta \to \infty$, the agent becomes a perfect optimizer, and as $\beta \to 0$,

the agent's actions become perfectly random). In the sequel, we assume that a given reward function will appropriately capture the role of β, so we omit β from our notation without loss of generality.

In the case of finite horizon planning, the partition function will be the sum of the exponentially weighted rewards for all feasible trajectories on MDP $\mathcal{M}$ of length no greater than the planning horizon. We denote this set of trajectories by $\Xi_{\mathcal{M}}$. Because adding constraints C modifies the set of feasible trajectories, we express this dependence as

$$Z(C) = \sum_{\xi \in \Xi_{\mathcal{M}}} e^{R(\xi)} \mathbb{1}^{\mathcal{M}^C}(\xi). \tag{6.7}$$

Assuming independence among demonstrated trajectories, the probability of observing a set $\mathcal{D}$ of N demonstrations is given by the product

$$P_{\mathcal{M}^C}(\mathcal{D}) = \frac{1}{Z(C)^N} \prod_{\xi \in \mathcal{D}} e^{R(\xi)} \mathbb{1}^{\mathcal{M}^C}(\xi). \tag{6.8}$$

Our goal is to maximize the demonstration probability given by (6.8). Because we take the reward function and demonstrations as given, our only available decision variable in this maximization is the constraint set C which alters the indicator $\mathbb{1}^{\mathcal{M}^C}$ and partition function $Z(C)$.

$$C^* = \operatorname*{argmax}_{C \in \mathcal{C}} P_{\mathcal{M}^C}(\mathcal{D}), \tag{6.9}$$

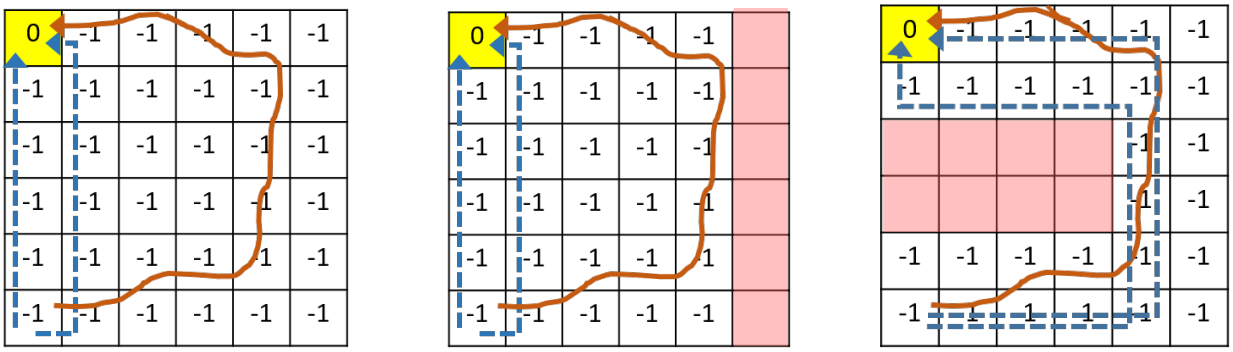
where $\mathcal{C} \subseteq 2^{S \times A}$ is the hypothesis space of possible constraints.

From the form of (6.8), it is clear that to solve (6.9), we must choose a constraint set that does not invalidate any demonstrated trajectory while simultaneously minimizing the value of $Z(C)$. Consider the set of trajectories that would be made *infeasible* by augmenting the MDP with constraint C, which we denote as $\Xi^-_{\mathcal{M}^C} = \{\xi \in \Xi_{\mathcal{M}} \mid \mathbb{1}^{\mathcal{M}^C}(\xi) = 0\}$. The value of $Z(C)$ is minimized when we maximize the sum of exponentiated rewards of these infeasible trajectories. Considering the form of the trajectory probability given by (6.6), we can see that this sum is proportional to the total probability of observing a trajectory from $\Xi^-_{\mathcal{M}^C}$ on the original MDP $\mathcal{M}$

$$\sum_{\xi \in \Xi^-_{\mathcal{M}^C}} e^{R(\xi)} \propto \sum_{\xi \in \Xi^-_{\mathcal{M}^C}} P_{\mathcal{M}}(\xi) = P_{\mathcal{M}}(\Xi^-_{\mathcal{M}^C}). \tag{6.10}$$

This insight leads us to the final form of the optimization

$$\begin{aligned} C^* = \operatorname*{argmax}_{C \in \mathcal{C}} \; & P_{\mathcal{M}}(\Xi^-_{\mathcal{M}^C}) \\ \text{s.t. } & \mathcal{D} \cap \Xi^-_{\mathcal{M}^C} = \emptyset \end{aligned}. \tag{6.11}$$



(a) Nominal MDP (b) Add constraint C_1 (c) Add constraint C_2

Figure 6.2: Selecting constraints to maximize demonstration likelihood. Trajectories that are likely to be observed on a given MDP are shown as dashed, angular arrows, and a provided demonstration is shown as a solid, curved arrow. Figure (a) shows these trajectories on a nominal MDP, and Figures (b) and (c) show the effects of adding two different constraints. While both C_1 and C_2 are possible constraints (i.e., they do not conflict with the demonstration), adding C_1 in (b) does little to align the expected trajectories with the demonstration. On the other hand, adding C_2 in (c) makes the original expected trajectories infeasible and causes the new expected trajectories to agree with the demonstration, greatly increasing the likelihood of the demonstration on this constrained MDP. This simple example illustrates how choosing constraints that eliminate likely trajectories on the nominal MDP will do more to increase the likelihood of demonstrations on the constrained MDP.

In order to solve (6.11), we must reason about the probability distribution of trajectories on the original MDP $\mathcal{M}$, then find the constraint C such that $\Xi^{-}_{\mathcal{M}^C}$ contains the most probability mass while not containing any demonstrated trajectories. Figure 6.2 provides a graphical representation of this reasoning. We highlight here that the fact that the chosen C must not conflict with *any* demonstration is an important condition: *all* provided demonstrations must perfectly respect a constraint in order for it to be learned, otherwise a less restrictive set of constraints may be learned instead. A promising future direction is to relax this requirement in order to learn about constraints that are generally respected by a set of demonstrations, without needing to first isolate just the successful, constraint-respecting demonstrations.

While equation (6.11) is derived for deterministic MDPs, if we can assume, as proposed by Ziebart, Maas, Bagnell, *et al.* [5], that for a given stochastic MDP, the stochastic outcomes have little effect on an agent's behavior and the partition function, then the solution to (6.11) will also approximate the optimal constraint selection for that MDP. However, in order to fully address the stochastic case, we would need to reformulate our approach based on maximum *causal* entropy [53]. We save this extension for future work.

Constraint Hypothesis Space

In order for the solutions to (6.11) to be meaningful, we must be careful with our choice of the constraint hypothesis space $\mathcal{C}$. For instance, if we let $\mathcal{C} = 2^{S \times A}$, then the optimal solution will always be to choose the most restrictive C to constrain *all* state-action pairs not observed in the demonstration set.

One approach to avoid this trivial solution is to use domain knowledge of the modeled system to restrict $\mathcal{C}$ to a library of plausible or common constraints. McPherson*, Scobee*, Menke, *et al.* [14] construct such a library by using reachability theory to calculate a family of likely unsafe sets.

We could also potentially address this problem by adding a regularization term to the optimization that would penalize constraints based on some notion of "size," which would encourage the selection of "smaller" constraints. The size of a constraint set could be defined by the number of *minimal constraints* that it contains. These minimal constraint sets constrain a single state, action, or feature, and were introduced in Section 6.2.2 as C_{s_i}, C_{a_i}, and C_{ϕ_i}, respectively. While penalizing this definition of size would effectively discourage overfitting, considering every possible combination of minimal constraints causes the hypothesis space to grow exponentially in the number of states, actions, and features of the MDP, which may make directly solving this formulation intractable.

Another approach, which avoids this combinatorial explosion, is to use the minimal constraint sets themselves, not their combinations, as our hypothesis space, and to select from among these constraints in an iterative, greedy manner. By iteratively selecting individual minimal constraint sets, and choosing a proper stopping condition, it is possible to gradually grow the full estimated constraint set and avoid overfitting to the demonstrations. It is this method that we will utilize in this work. Section 6.2.4 details our approach for selecting the most likely minimal constraint, and Section 6.2.5 details our approach for iteratively growing the estimated constraint set.

6.2.4 Probability Mass for Minimal Constraints

As detailed in Section 6.2.3, the most likely constraint set is the one whose eliminated trajectories $\Xi^{-}_{\mathcal{M}^C}$ have the highest probability of being demonstrated on the original, unconstrained MDP. Therefore, to find the most likely of the minimal constraints, we must find the expected proportion of trajectories which will contain any state or action, or accrue any feature. By using our augmented indicator feature map from (6.1), we can reduce this problem to only examine feature accruals.

Ziebart, Maas, Bagnell, *et al.* [5] present their forward-backward algorithm for calculating expected feature counts for an agent following a policy in the maximum entropy setting. This algorithm nearly suffices for our purposes, but it computes the expectation of the total number of times a feature will be accrued (i.e. how often will this feature be observed per trajectory), rather than the expectation of the number of trajectories that will accrue that feature at *any* point in time. To address this problem, we present a modified form

Algorithm 6.1 Feature Accrual History Calculation

Input: an MDP $\mathcal{M}$, a policy $\pi(a|s,t)$, a time horizon T

Output: expected feature accrual history $\widetilde{\Phi}_{[1,T]}$

/* Initialize state visitation and feature accrual history */

for $s \in S$ **do**

  $D_{s,0} \leftarrow D_0(s)$

  $\widetilde{\Phi}_{s,0} \leftarrow \mathbf{0}_{n_\phi \times 1}$

end for

/* Track feature accruals over the time horizon */

for $t \in [0,\ T-1]$ **do**

  for $s \in S$ **do**

    for $a \in A_s$ **do**

      /* New feature accruals */

      /* "$\odot$" denotes element-wise multiplication */

      $\Delta\widetilde{\Phi}_{s,t}(a) \leftarrow \widetilde{\phi}^{\mathbb{1}}(s,a) \odot \left(D_{s,t}\mathbf{1}_{n_\phi \times 1} - \widetilde{\Phi}_{s,t}\right)$

    end for

  end for

  for $s' \in S$ **do**

    $D_{s',t+1} \leftarrow \sum_{s \in S} \sum_{a \in A_s} D_{s,t}\pi(a|s,t)P(s'|s,a)$

    $\widetilde{\Phi}_{s',t+1} \leftarrow$
    $\sum_{s \in S} \sum_{a \in A_s} \left(\widetilde{\Phi}_{s,t} + \Delta\widetilde{\Phi}_{s,t}(a)\right)\pi(a|s,t)P(s'|s,a)$

  end for

  $\widetilde{\Phi}_{t+1} \leftarrow \sum_{s \in S} \widetilde{\Phi}_{s,t+1}$

end for

Return $\widetilde{\Phi}_{[1,T]}$

of the "forward" pass as Algorithm 6.1. Our algorithm tracks state visitations as well as feature accruals at each state, which allows us to produce the same maximum entropy distribution over trajectories as [5] while not counting additional accruals for trajectories that have already accrued a feature.

The input of Algorithm 6.1 includes the MDP itself, a time horizon, and a time-varying policy. This policy should capture the expected behavior of the demonstrator on the nominal MDP $\mathcal{M}$, and it can be computed via the "backward" part of the algorithm from [5]. The output of Algorithm 6.1, $\widetilde{\Phi}_{[1,T]}$, is an $n_\phi \times T$ array such that the t-th column $\widetilde{\Phi}_t$ is a vector whose i-th entry is the expected proportion of trajectories to have accrued the i-th feature by time t. In particular, the i-th element of $\widetilde{\Phi}_T$ is equal to $P_{\mathcal{M}}(\Xi^{-}_{\mathcal{M}^{C_i}})$, which allows us to

now directly select the most likely constraint according to (6.11).

6.2.5 Maximum-Coverage-Based Iterative Constraint Inference

When using minimal constraint sets as the constraint hypothesis space, it is possible that the most likely constraint still does not provide a satisfactory explanation for the demonstrated behavior. In this case, it can be beneficial to combine minimal constraints. If the task of solving (6.11) is framed as finding the combination of constraint sets that "covers" the most probability mass, then the problem becomes a direct analog for the classic maximum coverage problem. While this problem is known to be NP-hard, there exist a simple greedy algorithm with known suboptimality bounds [90].

Algorithm 6.2 Greedy Iterative Constraint Inference

Input: MDP $\mathcal{M}$, constraint hypothesis space $\mathcal{C}$,
empirical probability distribution $P_{\mathcal{D}}$, threshold $d_{D_{\mathrm{KL}}}$
Output: estimated constraint set $\widehat{C}^*$

$\widehat{C}^* \leftarrow \emptyset$
for $i \in [1,\ |\mathcal{C}|\,]$ **do**
  $C_i \leftarrow$ solution to (6.11) using $\mathcal{M}^{\widehat{C}^*}$, $\mathcal{C}$, and $\mathcal{D}$
  $\Delta_{D_{\mathrm{KL}}} = D_{\mathrm{KL}}\big(P_{\mathcal{D}} \,||\, P_{\mathcal{M}^{\widehat{C}^*}}\big) - D_{\mathrm{KL}}\big(P_{\mathcal{D}} \,||\, P_{\mathcal{M}^{\widehat{C}^* \cup C_i}}\big)$
  if $\Delta_{D_{\mathrm{KL}}} \leq d_{D_{\mathrm{KL}}}$ **then**
    break
  end if
  $\widehat{C}^* \leftarrow \widehat{C}^* \cup C_i$
end for
Return $\widehat{C}^*$

We present Algorithm 6.2 as our approach for adapting this greedy heuristic to solve the problem of constraint inference. At each iteration, we grow our estimated constraint set by augmenting it with the constraint set in our hypothesis space that covers the most *currently uncovered* probability mass. By analogy to the maximum coverage problem [90], we derive the following bound on the suboptimality of our approach.

Theorem 1. *Let $\mathcal{C}_{n_c}$ be the set of all constraints $\mathbf{C}_{n_c}$ such that $\mathbf{C}_{n_c} = \bigcup_{i=1}^{n_c} C_i$ for $C_i \in \mathcal{C}$, and let $\mathbf{C}^*_{n_c}$ be the solution to* (6.11) *using $\mathcal{C}_{n_c}$ as the constraint hypothesis space. It follows, then, that at the end of every iteration i of Algorithm 6.2,*

$$P\left(\Xi^-_{\mathcal{M}^{\widehat{C}^*}}\right) \geq \left(1 - \left(\frac{i-1}{i}\right)^i\right) P\left(\Xi^-_{\mathcal{M}^{\mathbf{C}^*_i}}\right).$$

Proof. The problem of finding $\mathbf{C}^*_{n_c}$ is analogous to solving the maximum coverage problem, presented in [90], where the set of elements to be covered is the set of trajectories

$\{\xi \mid \exists C \in \mathcal{C} : \xi \in \Xi^-_{\mathcal{M}^C}$ and $\mathcal{D} \cap \Xi^-_{\mathcal{M}^C} = \emptyset\}$ and the weight of each element ξ is $P_{\mathcal{M}}(\xi)$. Because Algorithm 6.2 constructs $\widehat{C}^*$ iteratively by taking the union of the previous value of $\widehat{C}^*$ and the set $C_i \in \mathcal{C}$ which solves (8), the value of $\widehat{C}^*$ at the end of the i-th iteration is analogous to the greedy solution of the maximum coverage problem with $n_c = i$. Therefore, we can directly apply the suboptimality bound for the greedy solution proven in [90] to arrive at our given bound on eliminated probability mass. ■

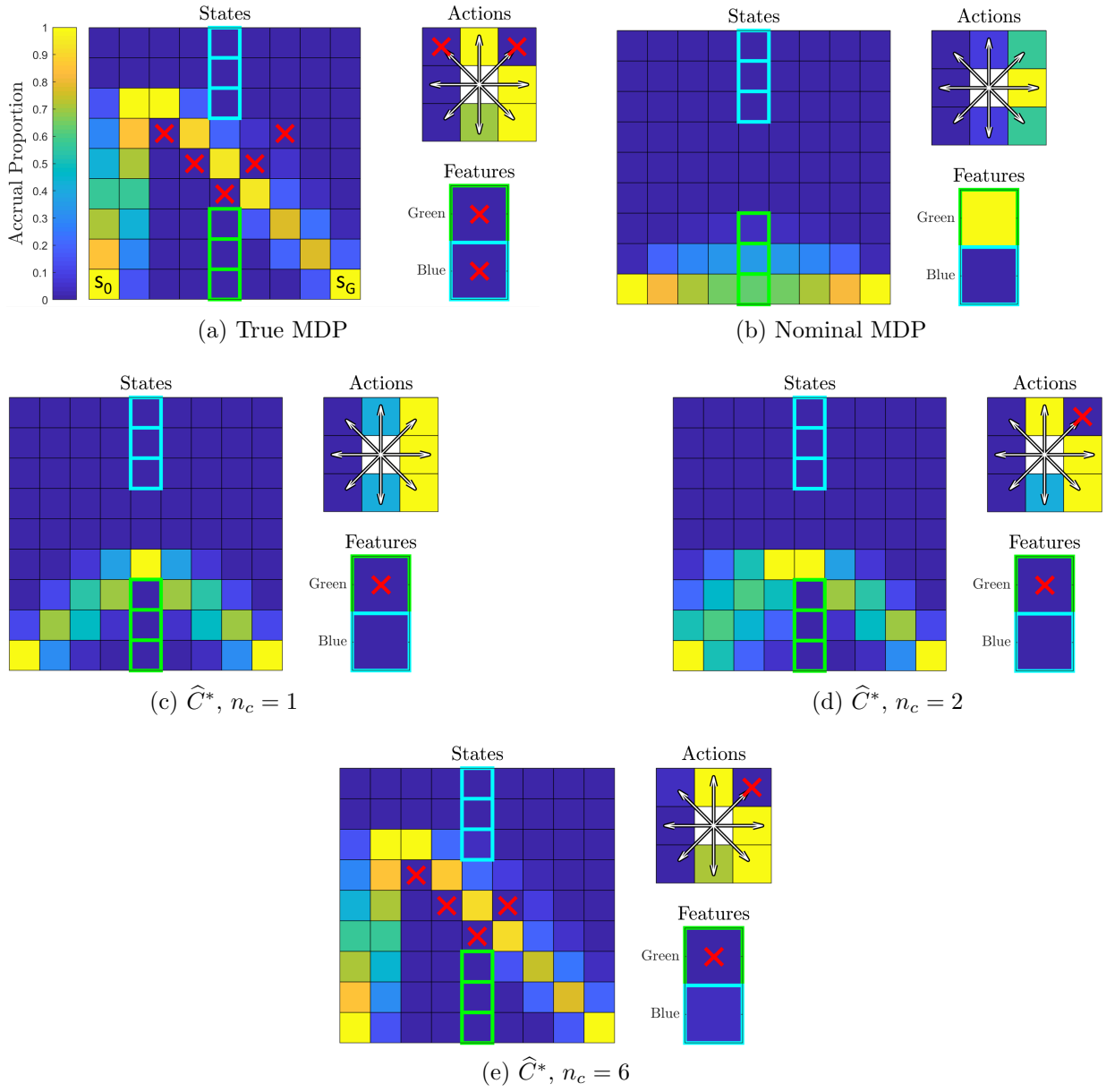
Rather than selecting the number of constraints n_c to be used ahead of time, we check a stopping condition to decide if we should continue to add constraints. Because we are attempting to maximize $P_{\mathcal{M}^C}(\mathcal{D})$, it might seem natural to use a probability-based stopping condition. However, choosing a stopping criterion based on probability is problematic because the probability of observing a set of demonstrations is dependent on the number of demonstrations in the set, even if each individual demonstration has the same probability. We instead base our stopping condition on KL divergence, which depends only on the distribution of trajectories in $\mathcal{D}$ and not on the number of demonstrations. The quantity $D_{\mathrm{KL}}\big(P_{\mathcal{D}} \,||\, P_{\mathcal{M}^{\widehat{C}^*}}\big)$ provides a measure of how well the distribution over trajectories induced by our inferred constraints, $P_{\mathcal{M}^{\widehat{C}^*}}$, agrees with the empirical probability distribution over trajectories observed in the demonstrations, $P_{\mathcal{D}}$. Using this KL divergence in the stopping condition actually preserves a link to the probability of the demonstration set, since the KL divergence will decrease monotonically as $P_{\mathcal{M}^C}(\mathcal{D})$ increases. The threshold parameter $d_{D_{\mathrm{KL}}}$ is chosen to avoid overfitting to the demonstrations, combating the tendency to select additional constraints that may only marginally better align our predictions with the demonstrations.

6.3 Examples

6.3.1 Synthetic Grid World

We consider the grid world MDP presented in Figure 6.3. The environment consists of a 9-by-9 grid of states, and the actions are to move up, down, left, right, or diagonally by one cell. The objective is to move from the starting state in the bottom-left corner (s_0) to the goal state in the bottom-right corner (s_G). Every state-action pair produces a distance feature, and the MDP reward is negative distance, which encourages short trajectories. There are additionally two more features, denoted green and blue, which are produced by taking actions from certain states, as shown in Figure 6.3.

The true MDP, from which agents generate trajectories, is shown in Figure 6.3a, including its constraints. The nominal, more generic MDP shown in Figure 6.3b is what we take as $\mathcal{M}$ for applying the iterative maximum likelihood constraint inference in Algorithm 6.2, with feature accruals estimated using Algorithm 6.1. While Figures 6.3c through 6.3e show the iteratively estimated constraints, which align with the true constraints, it is interesting to note that not all constraints present in the true MDP are identified. For instance, it is



(a) True MDP

(b) Nominal MDP

(c) $\widehat{C}^*$, $n_c = 1$

(d) $\widehat{C}^*$, $n_c = 2$

(e) $\widehat{C}^*$, $n_c = 6$

Figure 6.3: Algorithm performance on a synthetic grid world MDP. Each subfigure represents the MDP by showing (clockwise from left) its states, actions, and features. Each element is shaded according to the proportion of trajectories that are expected to accrue the respective augmented feature, computed via Algorithm 6.1. Constraints are marked with a red "X," and bright bounding boxes mark the green and blue feature-producing states. The result here are shown for a set of 100 demonstrations sampled according to the expectation for the True MDP (a). We begin with the nominal MDP shown in (b), and produce (c), (d), and (e) by applying Algorithm 6.2. Note that (c), (d), and (e) show the selections of feature, action, and state constraints, respectively.

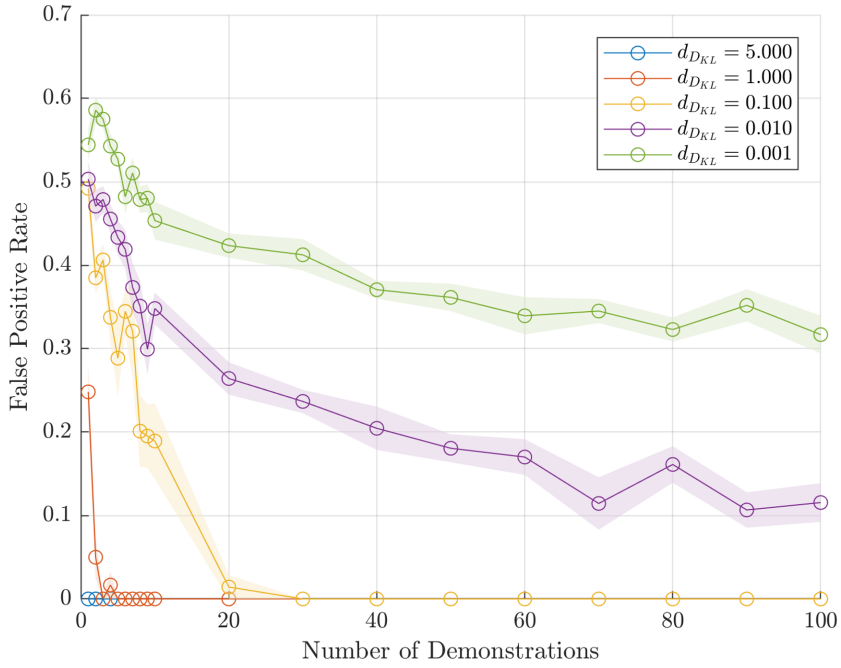
so unlikely that an agent would ever select the up-left diagonal action, that the fact that demonstrated trajectories did not contain that action is unsurprising and does not make that action an estimated constraint.

Figure 6.4 shows how the performance of our approach varies based on the number of available demonstrations and the selection for the threshold $d_{D_{\text{KL}}}$. The false positive rate shown in Figure 6.4a is the proportion of selected constraints which are not constraints of the true system. We can observe two trends in this data that we would expect. First, lower values of $d_{D_{\text{KL}}}$ lead to greater false positive rates since they allow Algorithm 6.2 to continue iterating and accept constraints that do less to align expectations and demonstrations. Second, having more demonstrations available provides more information and reduces the rate of false positives. Further, Figure 6.4b shows that more demonstrations also allows the behavior predicted by constraints to better align with the observations. It is interesting to note, however, that with fewer than 10 demonstrations and a very low $d_{D_{\text{KL}}}$, we may produce very low KL divergence, but at the cost of a high false positive rate. This phenomenon highlights the role of selecting $d_{D_{\text{KL}}}$ to avoid over-fitting. The threshold $d_{D_{\text{KL}}} = 0.1$ achieves a good balance of producing few false positives with sufficient examples while also producing lower KL divergences, and we used this threshold to produce the results in Figures 6.3 and 6.5.

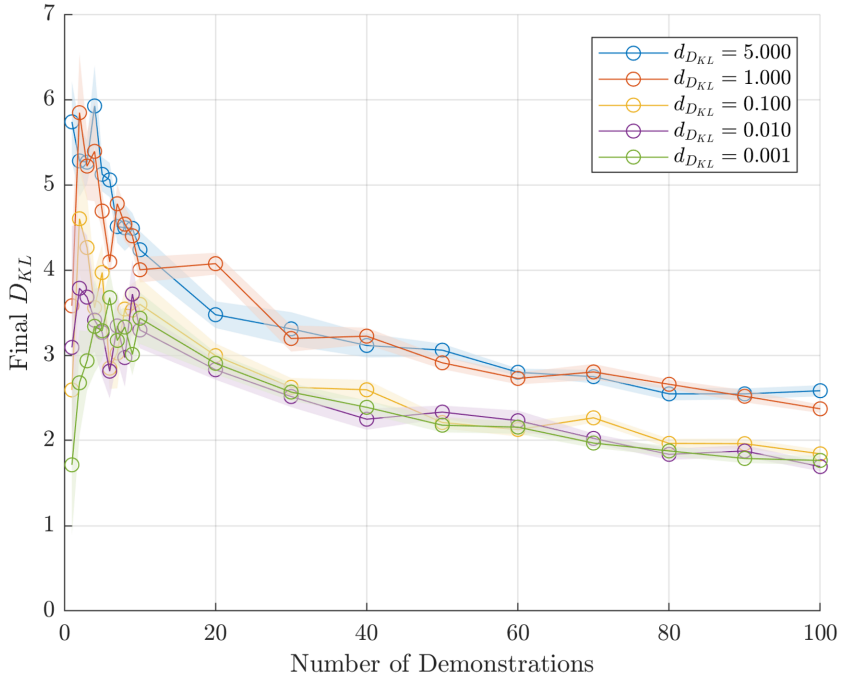
6.3.2 Human Obstacle Avoidance

In our second example, we analyze trajectories from humans as they navigate around an obstacle on the floor. We map these continuous trajectories into trajectories through a grid world where each cell represents a 1ft-by-1ft area on the ground. The human agents are attempting to reach a fixed goal state (s_G) from a given initial state (s_0), as shown in Figure 6.5. We performed MaxEnt IRL on human demonstrations of the task without the obstacle to obtain the nominal distance-based reward function. We restrict ourselves to estimating only state constraints, as we do not supply our algorithm with knowledge of any additional features in the environment and we assume that the humans' motion is unrestrained.

Demonstrations were collected from 16 volunteers, and the results of performing constraint inference are shown in Figure 6.5. Our method is able to successfully predict the existence of a central obstacle. While we do not estimate every constrained state, the constraints that we do estimate make all of the obstacle states unlikely to be visited. In order to identify those states as additional constraints, we would have to decrease our $d_{D_{\text{KL}}}$ threshold, which could also lead to more spurious constraint selections, such as the three shown in Figure 6.5a. We note, however, that as shown by Figures 6.5b and 6.5c, those spurious constraints actually do serve to better align our expectations with the human demonstrations. This result indicates that our algorithm for constraint inference is having the intended effect of maximizing demonstration likelihood, but that there may have been an unmodeled bias influencing the demonstrated human behavior.

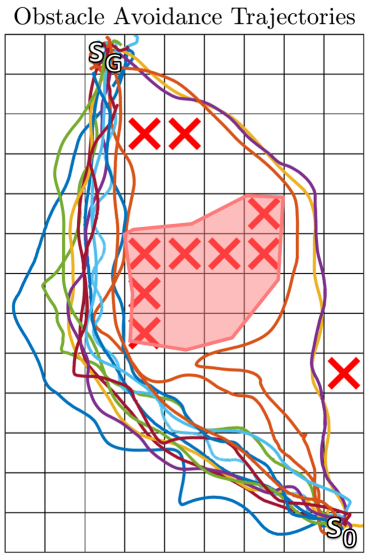


(a) False positives

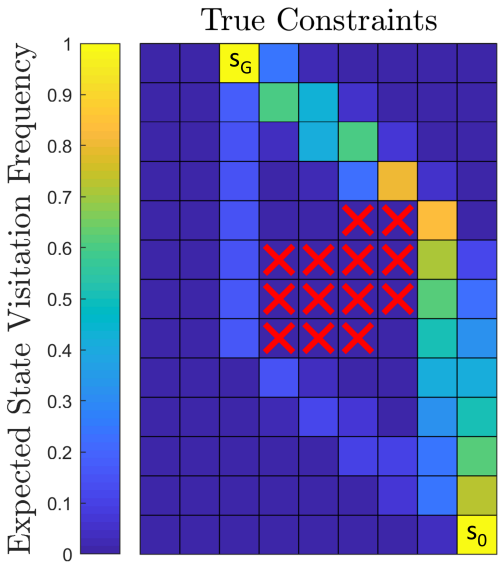


(b) D_{KL}

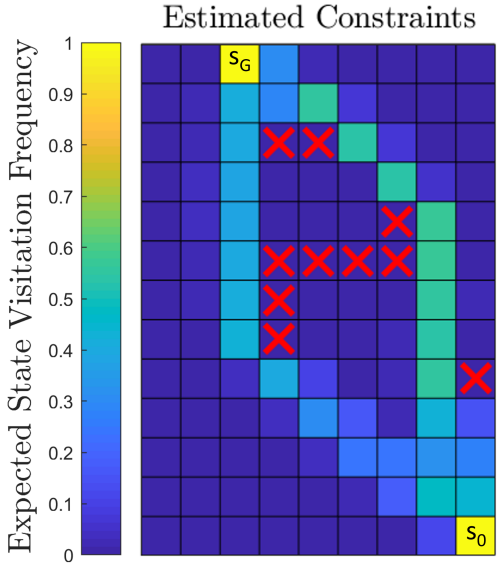
Figure 6.4: Algorithm performance on the synthetic grid world. Each data point represents the mean result of 10 independent trajectory draws, and the margins show ±1 standard error.



(a) Human Demonstrations



(b) Expectations with True Constraints



(c) Expectations with Estimated Constraints

Figure 6.5: In (a), we show the humans' trajectories overlaid on a grid world MDP. The shaded region represents an obstacle in the humans' environment, and the red "X"s mark the estimated constraints. In (b), the "X"s mark the true constraints, and the states are shaded according to their expected visitation frequencies with those constraints. The same is true of (c), but with respect to the estimated constraints. By shifting more probability mass to the left of the obstacle, the expectations under the estimated constraints actually better align with the demonstrations than the expectations under the true constraints. This discrepancy indicates that our algorithm is performing as intended, but that there may have been an unmodeled effect influencing human behavior.

6.4 Conclusion and Future Work

We have presented our novel technique for learning constraints from demonstrations. We improve upon previous work in constraint-learning IRL by providing a principled framework for identifying the *most likely* constraint(s), and we do so in a way that explicitly makes state, action, and feature constraints all directly comparable to one another. We believe that the numerical results presented in Section 6.3 are promising and highlight the usefulness of our approach.

Despite its benefits, one drawback of our approach is that the formulation is based on (6.6), which only exactly holds for deterministic MDPs. As mentioned in Section 6.2.3, we plan to investigate the use of a maximum *causal* entropy approach to address this issue and fully handle stochastic MDPs. Additionally, the methods presented here require *all* demonstrations to contain *no* violations of the constraints we will estimate. We believe that softening this requirement, which would allow reasoning about the likelihood of constraints that are occasionally violated in the demonstration set, may be beneficial in cases where trajectory data is collected without explicit labels of success or failure. Finally, the structure of Algorithm 6.1, which tracks the expected features accruals of trajectories over time, suggests that we may be able to reason about non-Markovian constraints by using this historical information to our advantage.

Overall, we believe that our formulation of maximum likelihood constraint inference for IRL shows promising results and presents attractive avenues for further investigation.

Part IV

Conclusions

Chapter 7

Future Directions

We are excited by the contributions that we have made toward integrating safety considerations into the IRL learning process and learning-based assistance. We believe that the approaches we present open up promising opportunities for further research, and we offer some possible future direction in this chapter.

Combining Haptic Feedback and Input-mixing Assistance

While we examine the differences between pure haptic feedback and pure input-mixing assistance in Chapter 4, we do not explore the possible benefits of combining the two approaches into a single learning-enabled assistance policy. As we mention in that analysis, this comparison was inspired by the user study in [12], which found that when intent inference was uncertain, users preferred a "timid" input-mixing assistance policy that gradually increased the level of assistance as confidence in intent prediction increased. It would be worthwhile to explore if a similar trade-off, with stronger haptic feedback under uncertain predictions and stronger input-mixing under confident predictions, could further enhance the user's experience of control and the safety of the system.

Performance vs. False Positives in Supervisor Safe Sets

As we discuss in Sections 5.1.4 and 5.3.3, by learning σ^2 in addition to μ, we are able to make predictions about the percentage of supervisor false positives that will be eliminated by avoiding keep-out sets at any given value threshold $\mu + z\sigma$, with $z \in \mathbb{R}$. Executing avoidance maneuvers at higher value thresholds causes robots to leave larger buffers between themselves and the keep-out set, which can lead to more circuitous and less efficient trajectories. If we assign a cost to each expected false positive, it would become possible to optimize for the trade-off between reducing false positives and increasing path efficiency. Such an analysis could shed light on the results from Section 5.3.2, where no significant preference was found between avoidance based on the Learned safe set and the Conservative safe set, which was less efficient but tended to produce fewer false positives.

Adaptive Supervisor Safe Sets

Once we begin using a learned safe set for robot control, the robots should never again violate that learned safe set (under nominal operation). However, the learned safe sets are based on human data collected at a specific point in time, and the approach assumes that human safety preferences are static. As humans become more experienced with a system and their understanding of its dynamics improves, could their preferences for safety become less conservative? This change in preference seems like it may be a natural consequence of increased familiarity with the system. Nonetheless, the question remains as to how we could update learned safe sets to become *less* conservative when the safety controller itself will prevent the robot from leaving the current learned safe set and inducing interventions based on the human's new safe set. In this case, the supervisor safe set technique would benefit from a modified formulation that allows for this type of adaptation.

Softening Constraint Learning

Our method for learning constraints from demonstration (Chapter 6) requires that all provided demonstrations respect a constraint in order for that constraint to be learnable. In a sense, we require all demonstration trajectories to be "labeled" as respecting constraints, and we assume that if any example trajectories did violate a relevant constraint then it was discarded and not provided as a demonstration. However, this approach may be restrictive, as we may not have access to labels to identify only the trajectories that respected the constraints of interest. We may benefit from integrating ideas in [84], where the authors are still able to infer task specifications from demonstrations, even in the presence of some demonstrations that may, unintentionally, violate a task specification. We note that while such an advance would soften the requirements of the learning process, we would still be learning hard constraints which are satisfied probabilistically by demonstrators.

Causal Entropy for Constraint Learning

The Maximum Likelihood Constraint Inference formulation in Chapter 6 is developed from the formulation of Maximum Entropy IRL [5], which adopts a noisily rational model of human decision making to assign a probability to each possible MDP trajectory. As the authors of [5] note, this formulation is only exactly correct for MDPs with deterministic transitions, since only in those cases is it possible for a human to directly "choose" a trajectory. While the Maximum Entropy formulation may still approximately describe behavior for nondeterministic MDPs, an exact solution requires introducing the notion of *casual* entropy [53], [54]. The Maximum Causal Entropy IRL formulation only allows actions to be conditioned on information observed up to the current time. This change in formulation means that the noisily rational probability distribution is over *policies*, rather than trajectories. Since our formulation for constraint inference is based so heavily on probability mass being assigned directly to trajectories, special care must be taken to successfully integrate causal entropy.

Simultaneously Learning Rewards and Constraints

The learning techniques that we present in Part III of this dissertation focus on learning safety constraints either without particular consideration for a reward function (Chapter 5) or with a nominal reward already acquired (Chapter 6). A natural question arises: rather than learning a reward and then learning constraints separately, is it possible to learn rewards and constraints simultaneously? Instead of considering the human's policy as being driven by either a reward-based value function (e.g., the expected sum of rewards R in Section 2.3.1) or a safety-based value function (e.g., minimum future distance l to the keep-out set in Section 2.1.2) we could model the human as optimizing some combination of these values. However, without any other assumptions, this problem is ill-posed: a sufficiently negative reward can lead to avoidance of keep-out sets, and maximizing distance from keep-out sets could also drive agents towards high-reward regions. Careful consideration must be given for possible additional restrictions or regularizations that could be applied to this joint optimization problem to disambiguate the two effects. We believe that this could be an illuminating direction for future investigations.

Chapter 8

Conclusion

In this dissertation, we have taken a step toward addressing the incorporation of safety into the context of Inverse Reinforcement Learning. This effort is part of a larger goal to ensure that new robotic systems are safe to use and deploy, which is a critical element to their acceptance and wide adoption. Because emerging robotics applications will involve human interaction in uncertain and dynamic environments, robust solutions will need to be learning-enabled in order to adapt themselves to changing and difficult-to-specify conditions. It is this connection that drives us to focus on ways to integrate safety into the learning framework and the application of that learned knowledge.

One approach that we explore in Chapter 6 is to learn about safety considerations directly from human demonstrations. By adapting the formulation of Maximum Entropy IRL, we are able learn *hard* constraints given demonstrations and a nominal reward function motivating the demonstrated behavior. The constraints we learn are in general over *features* of the environment, which allows this knowledge to generalize to new regions sharing those features. Learning hard constraints also offers an advantage over soft constraints (i.e. reward penalties), because whether an agent respects those constraints is not dependent on variations to other aspects of the reward function. We argue that this invariance is a key component of capturing information related to safety.

While using knowledge gained about safety to modify robot behavior, we must also consider how that behavior will impact human-robot interactions. As our work in Chapter 5 highlights, it is important that autonomous systems are not only physically safe, but that they also *appear* safe to any humans interacting with or observing them. The Supervisor Safe Sets formulation that we develop allows us to learn about how humans perceive the safety of a dynamical system and to use that information to align robot behavior with human expectations. Our user study results illustrate how robot motion can communicate safety and reduce confusion on the part of human observers. This effect is just one way in which designing a robotic system with the human experience in mind can yield benefits.

In Part II, we explore another technique for applying learned information to create safer human-robot interactions in shared control. Because learning and human intent inference will not always be perfect, it is important to provide assistance in a flexible manner that

allows the human to maintain control in situations where the automation's understanding may be incorrect or incomplete. We found that employing haptic feedback for learning-based assistance can be an effective method for delivering assistance while maintaining the human's sense of control and improving safety in cases where the automation misunderstands the human's intent.

All of these contributions are aimed towards a future where interactions with learning robotic agents are both frequent and safe. There is still plenty of work left to achieve that goal, and we highlight some promising future directions in Chapter 7. We are excited about all of the resources, energy, and creativity being devoted to the advancement of robotics and artificial intelligence, and we are thankful to be able to contribute to this rewarding field.

Bibliography

[1] A. Y. Ng and S. J. Russell, "Algorithms for inverse reinforcement learning," in *Proceedings of the Seventeenth International Conference on Machine Learning*, Morgan Kaufmann Publishers Inc., 2000, pp. 663–670.

[2] N. D. Ratliff, J. A. Bagnell, and M. A. Zinkevich, "Maximum margin planning," in *Proceedings of the 23rd international conference on Machine learning*, ACM, 2006, pp. 729–736.

[3] P. Abbeel and A. Y. Ng, "Apprenticeship learning via inverse reinforcement learning," in *Proceedings of the twenty-first international conference on Machine learning*, ACM, 2004, p. 1.

[4] J. Z. Kolter, P. Abbeel, and A. Y. Ng, "Hierarchical apprenticeship learning with application to quadruped locomotion," in *Advances in Neural Information Processing Systems*, 2008, pp. 769–776.

[5] B. D. Ziebart, A. L. Maas, J. A. Bagnell, and A. K. Dey, "Maximum entropy inverse reinforcement learning.," in *Aaai*, Chicago, IL, USA, vol. 8, 2008, pp. 1433–1438.

[6] M. Vasic and A. Billard, "Safety issues in human-robot interactions," in *2013 IEEE International Conference on Robotics and Automation*, 2013, pp. 197–204. DOI: 10.1109/ICRA.2013.6630576.

[7] ANSI/RIA, *ANSI/RIA R15.06-2012: Industrial Robots And Robot Systems - Safety Requirements*. American National Standards Institute / Robotics Industry Association, 2012.

[8] ISO/TC 299, *ISO/TS 15066:2016: Robots and robotic devices - Collaborative robots*. International Organization for Standardization, 2016.

[9] P. G. Griffiths and R. B. Gillespie, "Sharing control between humans and automation using haptic interface: primary and secondary task performance benefits," *Human Factors*, vol. 47, no. 3, pp. 574–590, 2005, PMID: 16435698. eprint: https://doi.org/10.1518/001872005774859944.

[10] B. A. C. Forsyth and K. E. Maclean, "Predictive haptic guidance: intelligent user assistance for the control of dynamic tasks," *IEEE Transactions on Visualization and Computer Graphics*, vol. 12, no. 1, pp. 103–113, 2006, ISSN: 1077-2626.

[11] M. Mulder, D. A. Abbink, M. M. van Paassen, and M. Mulder, "Haptic gas pedal support during visually distracted car following," *IFAC Proceedings Volumes*, vol. 43, no. 13, pp. 322 –327, 2010, 11th IFAC/IFIP/IFORS/IEA Symposium on Analysis, Design, and Evaluation of Human-Machine Systems, ISSN: 1474-6670.

[12] A. D. Dragan and S. S. Srinivasa, "A policy-blending formalism for shared control," *The International Journal of Robotics Research*, vol. 32, no. 7, pp. 790–805, 2013. eprint: https://doi.org/10.1177/0278364913490324.

[13] D. R. R. Scobee, V. R. Royo, C. J. Tomlin, and S. S. Sastry, "Haptic assistance via inverse reinforcement learning," in *2018 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, IEEE, 2018, pp. 1510–1517.

[14] D. L. McPherson*, D. R. R. Scobee*, J. Menke, A. Y. Yang, and S. S. Sastry, "Modeling supervisor safe sets for improving collaboration in human-robot teams," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018, pp. 861–868. DOI: 10.1109/IROS.2018.8593865, (*Equal contribution).

[15] D. R. R. Scobee and S. S. Sastry, "Maximum likelihood constraint inference for inverse reinforcement learning," in *International Conference on Learning Representations*, 2020. [Online]. Available: https://openreview.net/forum?id=BJliakStvH.

[16] E. A. Coddington and N. Levinson, *Theory of ordinary differential equations*. Tata McGraw-Hill Education, 1955.

[17] J. F. Fisac, A. K. Akametalu, M. N. Zeilinger, S. Kaynama, J. Gillula, and C. J. Tomlin, "A general safety framework for learning-based control in uncertain robotic systems," *IEEE Transactions on Automatic Control*, vol. 64, no. 7, pp. 2737–2752, 2019, ISSN: 2334-3303. DOI: 10.1109/TAC.2018.2876389.

[18] S. Bansal, M. Chen, S. Herbert, and C. J. Tomlin, "Hamilton-jacobi reachability: a brief overview and recent advances," in *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, 2017, pp. 2242–2253. DOI: 10.1109/CDC.2017.8263977.

[19] L. C. Evans and P. E. Souganidis, "Differential games and representation formulas for solutions of hamilton-jacobi-isaacs equations," *Indiana University Mathematics Journal*, vol. 33, no. 5, pp. 773–797, 1984, ISSN: 00222518, 19435258. [Online]. Available: http://www.jstor.org/stable/45010271.

[20] J. H. Gillula, G. M. Hoffmann, H. Huang, M. P. Vitus, and C. J. Tomlin, "Applications of hybrid reachability analysis to robotic aerial vehicles," *The International Journal of Robotics Research*, vol. 30, no. 3, pp. 335–354, 2011. eprint: https://doi.org/10.1177/0278364910387173.

[21] N. B. Sarter and D. D. Woods, "Team play with a powerful and independent agent: operational experiences and automation surprises on the airbus a-320," *Human Factors*, vol. 39, no. 4, pp. 553–569, 1997, PMID: 11536850. eprint: https://doi.org/10.1518/001872097778667997.

[22] J. Venrooij, D. A. Abbink, M. Mulder, M. M. van Paassen, M. Mulder, F. C. T. van der Helm, and H. H. Bülthoff, "A biodynamic feedthrough model based on neuromuscular principles," *IEEE Transactions on Cybernetics*, vol. 44, no. 7, pp. 1141–1154, 2014, ISSN: 2168-2267.

[23] D. T. McRuer, "Pilot-induced oscillations and human dynamic behavior," *NASA Technical Report*, Aug. 1995.

[24] D. A. Abbink, M. Mulder, and E. R. Boer, "Haptic shared control: smoothly shifting control authority?" *Cognition, Technology & Work*, vol. 14, no. 1, pp. 19–28, 2012, ISSN: 1435-5566.

[25] R. C. Goertz, "Manipulators used for handling radioactive materials," *Human factors in technology*, pp. 425–443, 1963.

[26] S. Javdani, S. S. Srinivasa, and J. A. Bagnell, "Shared autonomy via hindsight optimization," *Robotics: Science and Systems*, 2015. eprint: `http://www.roboticsproceedings.org/rss11/p32.pdf`.

[27] D. A. Abbink and M. Mulder, "Neuromuscular analysis as a guideline in designing shared control," in *Advances in haptics*, InTech, 2010.

[28] L. B. Rosenberg, "Virtual fixtures: perceptual tools for telerobotic manipulation," in *Proceedings of IEEE Virtual Reality Annual International Symposium*, 1993, pp. 76–82.

[29] T. M. Lam, H. W. Boschloo, M. Mulder, and M. M. van Paassen, "Artificial force field for haptic feedback in uav teleoperation," *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, vol. 39, no. 6, pp. 1316–1330, 2009, ISSN: 1083-4427.

[30] Z. Wang, K. Mlling, M. P. Deisenroth, H. B. Amor, D. Vogt, B. Schlkopf, and J. Peters, "Probabilistic movement modeling for intention inference in humanrobot interaction," *The International Journal of Robotics Research*, vol. 32, no. 7, pp. 841–858, 2013. DOI: `10.1177/0278364913478447`. eprint: `https://doi.org/10.1177/0278364913478447`. [Online]. Available: `https://doi.org/10.1177/0278364913478447`.

[31] E. Rehder and H. Kloeden, "Goal-directed pedestrian prediction," in *2015 IEEE International Conference on Computer Vision Workshop (ICCVW)*, 2015, pp. 139–147. DOI: `10.1109/ICCVW.2015.28`.

[32] J. F. Fisac, A. Bajcsy, S. L. Herbert, D. Fridovich-Keil, S. Wang, C. Tomlin, and A. D. Dragan, "Probabilistically safe robot planning with confidence-based human predictions," in *Robotics: Science and Systems XIV, Carnegie Mellon University, Pittsburgh, Pennsylvania, USA, June 26-30, 2018*, 2018. DOI: `10.15607/RSS.2018.XIV.069`. [Online]. Available: `http://www.roboticsproceedings.org/rss14/p69.html`.

[33] A. Bajcsy, D. P. Losey, M. K. OMalley, and A. D. Dragan, "Learning robot objectives from physical human interaction," in *Conference on Robot Learning (CORL)*, 2017, pp. 217–226.

[34] T. Ikeda, Y. Chigodo, D. Rea, F. Zanlungo, M. Shiomi, and T. Kanda, "Modeling and prediction of pedestrian behavior based on the sub-goal concept," *Robotics: Science and Systems VIII*, p. 137, 2013.

[35] J. Kofman, Xianghai Wu, T. J. Luu, and S. Verma, "Teleoperation of a robot manipulator using a vision-based human-robot interface," *IEEE Transactions on Industrial Electronics*, vol. 52, no. 5, pp. 1206–1219, 2005, ISSN: 1557-9948. DOI: 10.1109/TIE.2005.855696.

[36] Jian Shen, J. Ibanez-Guzman, Teck Chew Ng, and Boon Seng Chew, "A collaborative-shared control system with safe obstacle avoidance capability," in *IEEE Conference on Robotics, Automation and Mechatronics, 2004.*, vol. 1, 2004, 119–123 vol.1. DOI: 10.1109/RAMECH.2004.1438902.

[37] D. Xiao and R. Hubbold, "Navigation guided by artificial force fields," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ser. CHI '98, Los Angeles, California, USA: ACM Press/Addison-Wesley Publishing Co., 1998, pp. 179–186, ISBN: 0-201-30987-4.

[38] P. Aigner and B. McCarragher, "Human integration into robot control utilising potential fields," in *Proceedings of International Conference on Robotics and Automation*, vol. 1, 1997, 291–296 vol.1. DOI: 10.1109/ROBOT.1997.620053.

[39] C. Ton, Z. Kan, and S. S. Mehta, "Obstacle avoidance control of a human-in-the-loop mobile robot system using harmonic potential fields," *Robotica*, vol. 36, no. 4, 463483, 2018. DOI: 10.1017/S0263574717000510.

[40] M. Li, M. Ishii, and R. H. Taylor, "Spatial motion constraints using virtual fixtures generated by anatomy," *IEEE Transactions on Robotics*, vol. 23, no. 1, pp. 4–19, 2007, ISSN: 1552-3098.

[41] R. A. Ruddle and D. M. Jones, "Movement in cluttered virtual environments," *Presence*, vol. 10, no. 5, pp. 511–524, 2001, ISSN: 1054-7460. DOI: 10.1162/105474601753132687.

[42] P. Trautman, "Assistive planning in complex, dynamic environments: a probabilistic approach," in *2015 IEEE International Conference on Systems, Man, and Cybernetics*, 2015, pp. 3072–3078.

[43] M. Li and A. M. Okamura, "Recognition of operator motions for real-time assistance using virtual fixtures," in *11th Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems, 2003. HAPTICS 2003. Proceedings.*, 2003, pp. 125–131.

[44] J. Ren, R. V. Patel, K. A. McIsaac, G. Guiraudon, and T. M. Peters, "Dynamic 3-d virtual fixtures for minimally invasive beating heart procedures," *IEEE Transactions on Medical Imaging*, vol. 27, no. 8, pp. 1061–1070, 2008, ISSN: 0278-0062.

[45] 3D Systems, *Phantom Premium 1.5/6DOF Haptic Device*.

[46] R. E. Kalman, "When is a linear control system optimal?" *Journal of Basic Engineering*, vol. 86, no. 1, pp. 51–60, 1964.

[47] M. K. Ho, M. L. Littman, F. Cushman, and J. L. Austerweil, "Teaching with rewards and punishments: reinforcement or communication?" In *CogSci*, 2015.

[48] D. Amodei, C. Olah, J. Steinhardt, P. F. Christiano, J. Schulman, and D. Mané, "Concrete problems in AI safety," *CoRR*, vol. abs/1606.06565, 2016. arXiv: 1606.06565. [Online]. Available: http://arxiv.org/abs/1606.06565.

[49] E. T. Jaynes, "Information theory and statistical mechanics," *Physical review*, vol. 106, no. 4, p. 620, 1957.

[50] A. Boularias, J. Kober, and J. Peters, "Relative entropy inverse reinforcement learning," in *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, G. Gordon, D. Dunson, and M. Dudk, Eds., ser. Proceedings of Machine Learning Research, vol. 15, Fort Lauderdale, FL, USA: PMLR, 2011, pp. 182–189. [Online]. Available: http://proceedings.mlr.press/v15/boularias11a.html.

[51] S. Levine and V. Koltun, "Continuous inverse optimal control with locally optimal examples," in *Proceedings of the 29th International Coference on International Conference on Machine Learning*, ser. ICML12, Edinburgh, Scotland: Omnipress, 2012, 475482, ISBN: 9781450312851.

[52] C. Finn, S. Levine, and P. Abbeel, "Guided cost learning: deep inverse optimal control via policy optimization," in *Proceedings of The 33rd International Conference on Machine Learning*, M. F. Balcan and K. Q. Weinberger, Eds., ser. Proceedings of Machine Learning Research, vol. 48, New York, New York, USA: PMLR, 2016, pp. 49–58. [Online]. Available: http://proceedings.mlr.press/v48/finn16.html.

[53] B. D. Ziebart, "Modeling purposeful adaptive behavior with the principle of maximum causal entropy," PhD thesis, Carnegie Mellon University, 2010.

[54] B. D. Ziebart, J. A. Bagnell, and A. K. Dey, "Modeling interaction via the principle of maximum causal entropy," in *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ser. ICML'10, Haifa, Israel: Omnipress, 2010, pp. 1255–1262, ISBN: 978-1-60558-907-7. [Online]. Available: http://dl.acm.org/citation.cfm?id=3104322.3104481.

[55] R. Bellman, "A markovian decision process," *Journal of mathematics and mechanics*, pp. 679–684, 1957.

[56] K. Muelling, A. Venkatraman, J.-S. Valois, J. E. Downey, J. Weiss, S. Javdani, M. Hebert, A. B. Schwartz, J. L. Collinger, and J. A. Bagnell, "Autonomy infused teleoperation with application to brain computer interface controlled manipulation," *Autonomous Robots*, vol. 41, no. 6, pp. 1401–1422, 2017, ISSN: 1573-7527.

[57] L. V. Herlant, R. M. Holladay, and S. S. Srinivasa, "Assistive teleoperation of robot arms via automatic time-optimal mode switching," in *The Eleventh ACM/IEEE International Conference on Human Robot Interaction*, ser. HRI '16, Christchurch, New Zealand: IEEE Press, 2016, pp. 35–42, ISBN: 978-1-4673-8370-7.

[58] K. Hauser, "Recognition, prediction, and planning for assisted teleoperation of freeform tasks," *Autonomous Robots*, vol. 35, no. 4, pp. 241–254, 2013, ISSN: 1573-7527.

[59] E. You and K. Hauser, "Assisted teleoperation strategies for aggressively controlling a robot arm with 2d input," in *Proceedings of Robotics: Science and Systems*, Los Angeles, CA, USA, 2011.

[60] D. J. Kim, R. Hazlett-Knudsen, H. Culver-Godfrey, G. Rucks, T. Cunningham, D. Portee, J. Bricout, Z. Wang, and A. Behal, "How autonomy impacts performance and satisfaction: results from a study with spinal cord injured subjects using an assistive robot," *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, vol. 42, no. 1, pp. 2–14, 2012, ISSN: 1083-4427.

[61] S. Russell, "Learning agents for uncertain environments," in *Proceedings of the eleventh annual conference on Computational learning theory*, ACM, 1998, pp. 101–103.

[62] P. Abbeel, A. Coates, M. Quigley, and A. Y. Ng, "An application of reinforcement learning to aerobatic helicopter flight," in *Advances in neural information processing systems*, 2007, pp. 1–8.

[63] K. Mombaur, A. Truong, and J.-P. Laumond, "From human to humanoid locomotionan inverse optimal control approach," *Autonomous robots*, vol. 28, no. 3, pp. 369–383, 2010.

[64] J. Inga, F. Kpf, M. Flad, and S. Hohmann, "Individual human behavior identification using an inverse reinforcement learning method," in *2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, 2017, pp. 99–104.

[65] J. Van der Wal, "Discounted markov games: generalized policy iteration method," *Journal of Optimization Theory and Applications*, vol. 25, no. 1, pp. 125–138, 1978, ISSN: 1573-2878. DOI: 10.1007/BF00933260. [Online]. Available: https://doi.org/10.1007/BF00933260.

[66] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*, 1. MIT press Cambridge, 1998, vol. 1.

[67] H. Abdul Rahman, Y. Fai, and E. Su, "Analysis of human hand kinematics: forearm pronation and supination," *Journal of Medical Imaging and Health Informatics*, vol. 4, Apr. 2014.

[68] S.-L. Hwang, W. Barfield, T.-C. Chang, and G. Salvendy, "Integration of humans and computers in the operation and control of flexible manufacturing systems," *The International Journal of Production Research*, vol. 22, no. 5, pp. 841–856, 1984.

[69] C. J. Tomlin, "Towards automated conflict resolution in air traffic control1," *IFAC Proceedings Volumes*, vol. 32, no. 2, pp. 6564–6569, 1999.

[70] S. Javdani, H. Admoni, S. Pellegrinelli, S. S. Srinivasa, and J. A. Bagnell, "Shared autonomy via hindsight optimization for teleoperation and teaming," *The International Journal of Robotics Research*, vol. 37, no. 7, pp. 717–742, 2018.

[71] A. Jain, S. Sharma, T. Joachims, and A. Saxena, "Learning preferences for manipulation tasks from online coactive feedback," *The International Journal of Robotics Research*, vol. 34, no. 10, pp. 1296–1313, 2015.

[72] P. W. Battaglia, J. B. Hamrick, and J. B. Tenenbaum, "Simulation as an engine of physical scene understanding," *Proceedings of the National Academy of Sciences*, vol. 110, no. 45, pp. 18 327–18 332, 2013.

[73] K. A. Smith and E. Vul, "Sources of uncertainty in intuitive physics," *Topics in cognitive science*, vol. 5, no. 1, pp. 185–199, 2013.

[74] R. Shadmehr and F. A. Mussa-Ivaldi, "Adaptive representation of dynamics during learning of a motor task," *Journal of Neuroscience*, vol. 14, no. 5, pp. 3208–3224, 1994.

[75] R. M. Robinson, D. R. R. Scobee, S. A. Burden, and S. S. Sastry, "Dynamic inverse models in human-cyber-physical systems," in *Micro-and Nanotechnology Sensors, Systems, and Applications VIII*, International Society for Optics and Photonics, vol. 9836, 2016, p. 98361X.

[76] J. F. Fisac, M. Chen, C. J. Tomlin, and S. S. Sastry, "Reach-avoid problems with time-varying dynamics, targets and constraints," in *Proceedings of the 18th international conference on hybrid systems: computation and control*, ACM, 2015, pp. 11–20.

[77] 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," in *Control Conference (ECC), 2015 European*, IEEE, 2015, pp. 3304–3309.

[78] S. L. Herbert, M. Chen, S. Han, S. Bansal, J. F. Fisac, and C. J. Tomlin, "Fastrack: a modular framework for fast and guaranteed safe motion planning," in *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, 2017, pp. 1517–1522. DOI: 10.1109/CDC.2017.8263867.

[79] G. M. Hoffmann and C. J. Tomlin, "Decentralized cooperative collision avoidance for acceleration constrained vehicles," in *Decision and Control, 2008. CDC 2008. 47th IEEE Conference on*, IEEE, 2008, pp. 4357–4363.

[80] A. K. Akametalu and C. J. Tomlin, "Temporal-difference learning for online reachability analysis," in *Control Conference (ECC), 2015 European*, IEEE, 2015, pp. 2508–2513.

[81] C. L. Baker, J. B. Tenenbaum, and R. R. Saxe, "Goal inference as inverse planning," in *Proceedings of the Annual Meeting of the Cognitive Science Society*, vol. 29, 2007.

[82] I. M. Mitchell, "A toolbox of level set methods," *Dept. Comput. Sci., Univ. British Columbia, Vancouver, BC, Canada, http://www. cs. ubc. ca/˜ mitchell/ToolboxLS/toolboxLS. pdf, Tech. Rep. TR-2004-09*, 2004.

[83] D. Vasquez, B. Okal, and K. O. Arras, "Inverse reinforcement learning algorithms and features for robot navigation in crowds: an experimental comparison," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, 2014, pp. 1341–1346.

[84] M. Vazquez-Chanlatte, S. Jha, A. Tiwari, M. K. Ho, and S. Seshia, "Learning task specifications from demonstrations," in *Advances in Neural Information Processing Systems*, 2018, pp. 5367–5377.

[85] M. Pardowitz, R. Zollner, and R. Dillmann, "Learning sequential constraints of tasks from user demonstrations," in *5th IEEE-RAS International Conference on Humanoid Robots, 2005.*, IEEE, 2005, pp. 424–429.

[86] C. Pérez-D'Arpino and J. A. Shah, "C-learn: learning geometric constraints from demonstrations for multi-step manipulation in shared autonomy," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2017, pp. 4058–4065.

[87] G. Subramani, M. Zinn, and M. Gleicher, "Inferring geometric constraints in human demonstrations," in *Conference on Robot Learning*, 2018, pp. 223–236.

[88] G. Chou, D. Berenson, and N. Ozay, "Learning constraints from demonstrations," in *Springer Proceedings in Advanced Robotics (SPAR) series on 2018 Workshop on the Algorithmic Foundations of Robotics (WAFR 2018)*, Springer, 2018.

[89] M. Kalakrishnan, P. Pastor, L. Righetti, and S. Schaal, "Learning objective functions for manipulation," in *2013 IEEE International Conference on Robotics and Automation*, 2013, pp. 1331–1336. DOI: 10.1109/ICRA.2013.6630743.

[90] D. S. Hochbaum and A. Pathria, "Analysis of the greedy approach in problems of maximum k-coverage," *Naval Research Logistics (NRL)*, vol. 45, no. 6, pp. 615–627, 1998.