Bridging Safety and Learning in Human-Robot Interaction

by

Andrea V Bajcsy

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

Summer 2022

Bridging Safety and Learning in Human-Robot Interaction

Abstract

Bridging Safety and Learning in Human-Robot Interaction

by

Andrea V Bajcsy

Doctor of Philosophy in Engineering - Electrical Engineering and Computer Sciences

University of California, Berkeley

Professor Anca Dragan, Co-chair

Professor Claire Tomlin, Co-chair

From autonomous cars to systems operating in people's homes, robots must interact with humans. What makes this hard is that human behavior–especially when interacting with other agents–is vastly complex, varying between individuals, environments, and over time. A modern approach to deal with this problem is to rely on data and machine learning throughout the design process and deployment to build and refine models of humans. However, by blindly trusting their data-driven human models, robots might confidently plan unsafe behaviors around people, resulting in anything from miscoordination to potentially even dangerous collisions.

This dissertation aims to lay the foundations for formalizing and ensuring safety in human-robot interaction, particularly when robots *learn from* and *about* people. It discusses how treating robot learning algorithms as dynamical systems driven by human data enables safe human-robot interaction. We first introduce a Bayesian monitor which infers online if the robot's learned human model can evolve to well-explain observed human data. We then discuss how a novel, control-theoretic problem formulation enables us to formally quantify what the robot could learn online from human data and how quickly this learning could be achieved. Coupling these ideas with robot motion planning algorithms, we demonstrate how robots can safely and automatically adapt their behavior based on how trustworthy their learned human models are. This thesis ends by taking a step back and raising the question: *"What is the 'right' notion of safety when robots interact with people?"* and discusses opportunities for how rethinking our notions of safety can capture more subtle aspects of human-robot interaction.

To maminka and ocko.

Contents

List of Figures

List of Tables

Acknowledgments

When I was leaving to start my PhD, my dad hugged me goodbye and told me, "The perseverant get PhDs." When I look back at the past six years of graduate school, it is clear to me that being perseverant is straightforward—sometimes even easy—when surrounded by a community of inspiring and supportive people.

First, I would like to thank my co-advisors, Anca Dragan and Claire Tomlin.

I am incredibly lucky that Anca took a chance on me and let me be one of her first PhD students. Anca was instrumental in showing me how to wade through the uncertainty of research and academia. One of the very first things she taught me was how to identify a research project's "key insight", a lesson that I take with me to every new project. Her unbounded enthusiasm and desire to deeply understand why a method works have been a constant source of energy in my PhD. I am also grateful for our shared love of graphic design and for Anca taking me to an Edward Tufte lecture in San Francisco during my first year. I still think fondly of that lecture when making figures for a talk or paper.

To Claire, I owe my love of dynamical systems and, of course, Hamilton-Jacobi reachability. Claire's *221A: Linear Systems Theory* course was one of the first classes I took at Berkeley and, despite being thoroughly overwhelmed, I immediately loved it. I proceeded to take her entire controls course series. Claire's intellectual curiosity and optimism in the face of hard research challenges has always been infectious. Even on days when I would enter our research meetings dejected, I would always leave feeling reassured and capable of persisting. Unfortunately, succinctly summarizing all the ways in which Anca and Claire have shaped me as a researcher, mentor, and teacher in just a few sentences is futile. All I can hope to do is help others in the ways that they helped me.

I would also like to thank my thesis committee and the many faculty who supported me professionally: thank you to Ben Recht, George Pappas, Ken Goldberg, Marcia O'Malley, Marco Pavone, Maya Cakmak, Shankar Sastry, and Wolfram Burgard for your invaluable guidance.

I have been fortunate to be a part of not one, but two amazing lab groups: the InterACT Lab and the Hybrid Systems Lab. The InterACT lab was my first home, and although it has grown and change a lot throughout the years, it has always preserved a sense of joy, community, and lightheartedness. In particular, thank you to Andreea Bobu and Ellis Ratner for commiserating about the pains of robot experiments and always being there to brainstorm another harebrained research idea. The Hybrid Systems Lab was my second home, and it has always been full of inspiring people. In particular, I want to thank the *quadcopter gang*, Jaime Fisac, Sylvia Herbert, and David Fridovich-Keil, for making even the most tiring robot experiments insanely fun. I have never laughed so hard when trying to debug why $P(x_H^t \mid x_H^0) = -100,000,000$. I also want to thank Somil Bansal for teaching me the ins-and-outs of reachability, discussing robotics while eating plenty of Zampanos, and for agreeing that all humans should move at 6 m/s.

I have also been fortunate to collaborate with inspiring researchers outside of the InterACT and Hybrid Systems labs. Thank you to Boris Ivanovic, Edward Schmerling, and

Karen Leung[1] for bringing so much laughter, meme-culture, and exciting conversations to my NVIDIA internship. I also want to thank Dylan Losey for working with me on my very first PhD project and never failing to have something goofy to say. Last but not least, thank you to Thomas (Ran) Tian for giving me the opportunity to collaborate with you; you are an absolute joy to work with! I also want to thank the junior students—Anand Siththaranjan, Eli Bronstein, Regina Wang, Sampada Deglurkar, and Steven Wang—for trusting in me as a mentor.

I am so lucky to have an amazing group of friends with whom I travel, commiserate, and (most importantly) grab some boba with. Thank you to Amelia Bateman, Anna Rubakhina, Anusha Nagabandi, Carolyn Matl, Eric Mazumdar, Esther Rolf, Gautam Gunjala, Nick Antipa, Stan Smith, Tselil Schramm, Tynan McAuley, Tyler Westenbroek, Matt Lentz, and Matt Matl.

I also owe a huge thanks to my partner, Melvin, whose positive outlook on life and tendency towards excellence is truly inspiring. I am continuously amazed at his unwavering support in me, through the easy times and the tough ones. His cooking skills have also never waivered, and I'm so lucky that I can count on him to whip up the perfect meal to lift my spirits. I am also so grateful for my dog Cheerio, for never failing to make me laugh with her goofy antics.

Last but not least, I want to thank my parents, my brother, my grandparents, aunts, uncles, and cousins. I would not be the person I am today without each and every one of you. In particular, I want to say thank you to my mom and dad. I am truly lucky to have your steadfast support no matter what I do in life. *Dakujem.*

[1] I'm still looking forward to the day we travel to Australia together and eat *waaay* too much.

Chapter 1

Introduction

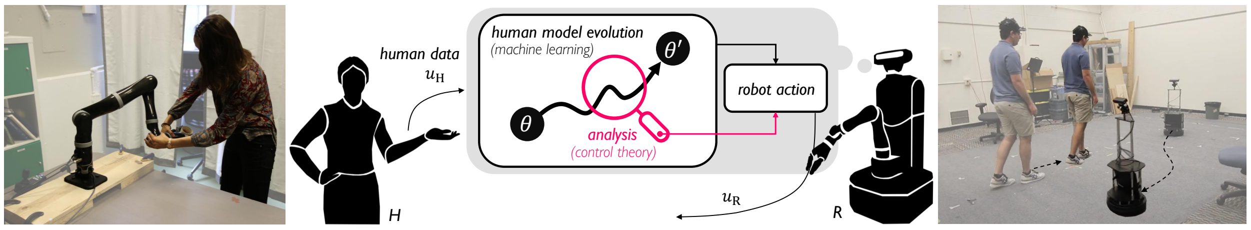


Figure 1.1: *Center:* By treating robot learning algorithms as dynamical systems driven by human data, this thesis unites traditionally disparate control-theoretic and machine learning methods for safe HRI. *Sides:* This work is grounded through robot experiments with robotic manipulators and autonomous vehicles.

From autonomous cars driving driving around cities to assistive robots helping out in the home, robots need to interact with people. Interaction requires robots to *learn from* and *about* people: where will a pedestrian walk to? How does a person want their valuables cleaned up in the home? How can an autonomous vehicle safely get to its destination while making the passenger feel comfortable? To build models of people, robots are increasingly relying on data and machine learning throughout the design process and during deployment. For example, using a learned human model trained on a huge dataset of human driving behavior, an autonomous car can predict likely maneuvers another car could do by simply observing a brief history of the human's actions.

Unfortunately, there is a fundamental and persistent challenge with human-robot interaction: human behavior is vastly complex, varying between individuals, environments, and over time. This means that when robots are deployed, they will encounter human behaviors that are unexpected and that they never saw during their design or training. For example, an autonomous car driving in San Francisco may encounter a pedestrian playing soccer on the side of the road, or a human who decides to intentionally block oncoming traffic. By blindly trusting their data-driven human models, robots can confidently plan unsafe behaviors, resulting in anything from miscoordination to head-on collisions. This exposes an underlying tension in human-robot interaction: robots need to learn about people in order to effectively interact with them, but these robots also need to be safe. In this thesis, we focus on precisely this tension, and outline some promising steps towards ensuring safety in human-robot interaction, particularly when robots are *learning from* and *about* people.

This dissertation is rooted in ideas from control theory—specifically reachability analysis—which mathematically characterizes how dynamical systems evolve and reach desired states. Classic physics-based reachability analysis has enabled robots like drones to avoid collision with obstacles, efficiently reach desired goals, and be robust to external disturbances like wind. Upon closer inspection, we can see that human-robot interaction has parallel objectives: robots should avoid colliding with people, efficiently learn a user's desired goals and preferences, and be robust to adversarial human feedback. However, now the robot's decisions, beliefs about the human's intent, and understanding of the task are all affected by the human behavior the robot observes. In other words, *when robots interact with people, it is not just physical state that evolves, but also the robot's learning.*

This leads us to the core idea of this dissertation:

> *safe interaction requires treating robot learning algorithms as dynamical systems whose evolution is "controlled" by human data.*

This control-theoretic perspective on the interplay between machine learning and human-robot interaction advances safety in traditional collision-avoidance scenarios and also unlocks novel research directions on safety related to robot learning capability, alignment with human values, and ability to optimize for human preferences (center, Fig. 1.1). We will demonstrate how to develop novel frameworks for analyzing how data-driven human models will evolve and derive theoretically rigorous and practical robot algorithms for safe interaction with people. Importantly, throughout this dissertation we will evaluate our novel methods through robotic hardware experiments with real human participants.

1.1 Thesis Overview and Contributions

This thesis focuses on ensuring safety for a class of human-robot interactions: robots learning online about human behavior in order to avoid collisions with people. In Chapter 2 we will introduce the necessary mathematical background to grasp the ideas in this dissertation. Specifically, we summarize key concepts from dynamical systems, optimal control, game theory, single-agent and multi-agent safety analysis, inverse reinforcement learning, and human motion prediction. The following chapters outline the key contributions on unifying safety and learning in human-robot interaction contexts like autonomous driving, drone navigation, ground robots, and assistive manipulators. See Fig. 1.2 for a visual representation of how this thesis fits into the broader context of problems at the intersection of safety, learning, and human-robot interaction.

Part I: Safe Robot Navigation Despite Imperfect Human Models. One of the most difficult challenges in robot navigation is to account for the behavior of humans. Commonly,

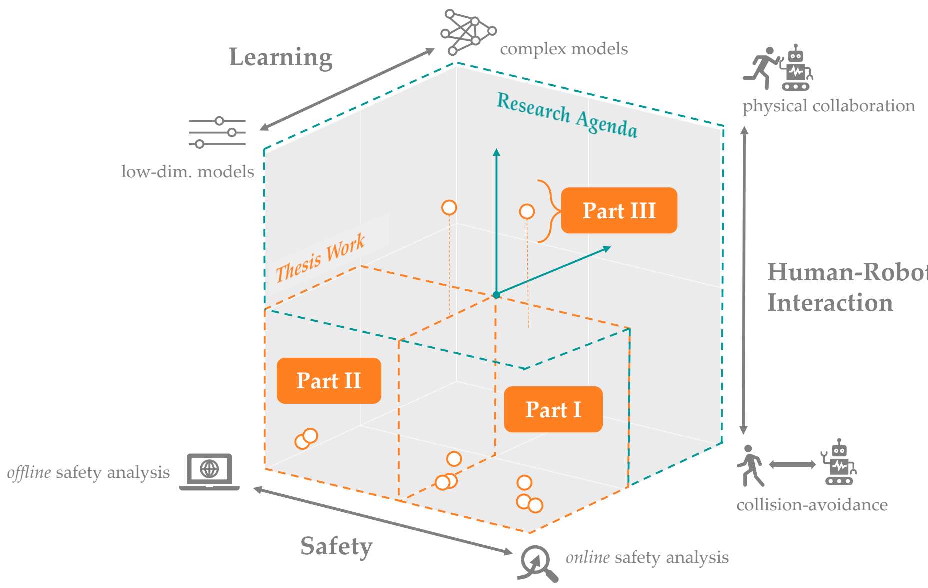


Figure 1.2: Thesis overview illustration, contextualizing this thesis in the broader space of questions concerning safety, learning, and human-robot interaction.

practitioners employ predictive models to reason about where humans will move. Though there has been much recent work in building predictive models, no model is ever perfect: a human can always move unexpectedly, in a way that is not predicted or not assigned sufficient probability. In such cases, the robot may plan trajectories that appear safe but, in fact, lead to collision. Chapter 3 proposes that rather than trust a model's predictions blindly, the robot should use the model's current predictive accuracy to inform the degree of confidence in its future predictions. This model confidence inference allows us to generate probabilistic motion predictions that exploit modeled structure when the structure successfully explains human motion, and degrade gracefully whenever the human moves unexpectedly. We accomplish this by maintaining a Bayesian belief over a single parameter that governs the variance of our human motion model. We couple this prediction algorithm with a recently proposed robust motion planner and controller to guide the construction of robot trajectories that are, to a good approximation, collision-free with a high, user-specified probability. In Chapter 4 we extend confidence-awareness to *game-theoretic* human models, enabling robots like autonomous cars to smoothly shift between collaborative interaction models, and safeguarding against worst-case adversarial human driving behavior when the interaction model degrades. Evaluations in simulated human-robot scenarios and ablation studies demonstrate that imbuing safety monitors with confidence-aware game-theoretic models enables both safe and efficient human-robot interaction. Unfortunately, modelling interaction effects (with games or otherwise) comes at a large computational cost, so it is often desirable to make simplifying assumptions at the expense of model fidelity. Chapter 5 introduces how model confidence-awareness enables multi-human, multi-robot planning algorithms to be assuredly deployed at scale by reasoning about observed behavior as deviation from simplified, tractable models (e.g., pairwise predictions). Finally, Chapter 6 proposes a different, but complementary, approach to "robustifying" predictive human models. We combine ideas from robust control and intent-driven human modelling to formulate a novel human motion predictor which provides robustness against miss-calibrated stochastic human models. Our approach predicts the human states by trusting the intent-driven model to decide only which human actions are completely unlikely. We then safeguard against all likely enough actions, much like a robust control predictor. We demonstrate in simulation and hardware how robots can behave safely around people without being overly conservative, even when relying on poorly calibrated human models.

Part II: Formalizing Safety Analysis of Adaptive Human Models. Predictive human models often need to adapt their parameters online from human data. For example, in Part I, the inferred model confidence parameter adapts the conservativness of the human predictions. This raises previously ignored safety-related questions for robots relying on these models such as what the model could learn online and how quickly could it learn it. For instance, when will the robot have a confident estimate in a nearby human's goal? Or, what parameter initializations guarantee that the robot can learn the human's preferences in a finite number of observations? To answer such analysis questions, Chapter 7 introduces the idea of modelling the robot's learning algorithm as a dynamical system where the state is the current model parameter estimate and the control is the human data the robot observes. This enables us to leverage tools from reachability analysis and optimal control to compute the set of hypotheses the robot could learn in finite time, as well as the worst and best-case times it takes to learn them. We demonstrate the utility of our analysis tool in four human-robot domains, including autonomous driving and indoor navigation.

Part III: Safety for Human-Robot Interaction Beyond Collision-Avoidance. In this final part, we move away from the navigation domain and ask "How can we formalize safety in *physical* human-robot interaction?" First, in Chapter 8 we recognize that physical human-robot interaction (pHRI) is often intentional – the human intervenes on purpose because the robot is not doing the task correctly. In this work, we argue that when pHRI is intentional it is also informative: the robot can leverage interactions to learn how it should complete the rest of its current task even after the person lets go. We formalize pHRI as a dynamical system, where the human has in mind an objective function they want the robot to optimize, but the robot does not get direct access to the parameters of this objective – they are internal to the human. Within our proposed framework human interactions become observations about the true objective. We introduce approximations to learn from and respond to pHRI in real-time. We recognize that not all human corrections are perfect: often users interact with the robot noisily, and so we improve the efficiency of robot learning from pHRI by reducing unintended learning. Finally, we conduct simulations and user studies on a robotic manipulator to compare our proposed approach to the state-of-the-art. Our results indicate that learning from pHRI leads to better task performance and improved human satisfaction. This research (and related prior work) lays the groundwork for how robots can use human input—like demonstrations or corrections—to learn intended objectives. However, these techniques assume that the human's desired objective already exists within the robot's hypothesis space. In reality, this assumption is often inaccurate: there will always be situations where the person might care about aspects of the task that the robot does not know about. Without this knowledge, the robot cannot infer the correct objective. Hence, when the robot's hypothesis space is *misspecified*, even methods that keep track of uncertainty over the objective fail because they reason about which hypothesis might be correct, and not whether *any* of the hypotheses are correct. This is a new type of safety concern, since the robot may behave in incorrect ways due to its inability to learn the intended objective. For example, we discovered that after consistently misinterpreting user feedback during a household cleaning task, the robot erroneously learns to move coffee mugs at an angle, resulting in spilled coffee and miscoordination. In Chapter 9, we posit that the robot should reason explicitly about how well it can explain human inputs given its hypothesis space and use that situational confidence to inform how it should incorporate human input. We demonstrate our method on a 7 degree-of-freedom robot manipulator in learning from two important types of human input: demonstrations of manipulation tasks, and physical corrections during the robot's task execution.

Chapter 2

Background and Preliminaries

2.1 Dynamical Systems

In this dissertation, we are concerned with how *robots* can safely interact with *humans*. Autonomous cars will coordinate with human-driven cars and pedestrians, assistive robotic manipulators will collaborate with end-users to perform daily living tasks, and autonomous quadrotors will fly through neighborhoods to deliver packages to peoples' homes. Understanding how humans and robots interact requires studying how these *systems* evolve over time. Where will a pedestrian walk to in the next five seconds? How should a robotic manipulator grab a cup from the shelf and hand it to the person?

In order to start tackling these questions, we need a unifying mathematical language to describe the evolution of autonomous systems over time (whether they be human or robotic). *Dynamical systems* theory is one such unifying framework. This branch of mathematics enables us to model, analyze, and predict the behavior of complex systems whose *state* evolves as a function of time. By *state*, we mean a minimum required set of variables that capture core properties of our system; e.g., the state of a person walking around a room should at least include their x- and y-position in the room. In the context human-robot interaction, our systems of interest will also evolve as a function of *control inputs*. These control inputs can be anything from high-level discrete decisions (e.g., at an intersection, a driver can choose to turn Left, Right, go Straight, or Stop), to lower-level continuous signals (e.g., commanding a robot's wrist actuator to rotate by $30°$ for three seconds). The choice of discrete or continuous representations of the control inputs, state-space, or time are all design decisions that an engineer must choose carefully given their application domain.

In this chapter, we will begin with dynamical systems that are continuous in time, state, and control, and later provide an overview of the other variants, such as discrete-time systems. We will also primarily focus on *deterministic* dynamical systems models, but we will briefly touch on *stochastic* dynamical systems models that are widespread in artificial intelligence (AI), reinforcement learning (RL), and human motion prediction.

Aside on Terminology. *The controls and artificial intelligence (AI) communities have long been interested in a similar problem: the design of autonomous systems. Despite a common interest, due to their distinct histories, the control theory and AI communities have evolved separate terminology for the same concepts. In this thesis, we draw upon ideas from both communities, and will use the terminology interchangeably. For clarity, below is a small table summarizing equivalent terms.*

Control Theory	**AI**
System	Environment
Dynamics (f)	Transition (T or P)
State (x)	State (s)
Control input (u)	Action (a)
Control law ($\mathbf{u}(\cdot)$)	Policy ($\pi(\cdot)$)
Output (y)	Observation (O or z)
Cost (J or C)	Reward (R)
Minimize (cost)	Maximize (reward)

Table 2.1: Terminology equivalence between the controls and AI communities.

2.1.1 Continuous-Time Dynamical Systems

Let's make these ideas about systems more mathematically concrete. Consider a dynamical system whose continuous state is $x \in \mathbb{R}^n$ and whose evolution is governed by the ordinary differential equation (ODE):

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

where the *control input* is $u \in \mathcal{U} \subseteq \mathbb{R}^m$ and the flow field is $f : \mathbb{R}^n \times \mathcal{U} \to \mathbb{R}^n$. We often assume that the control set $\mathcal{U}$ is compact, since in practice our systems can only exert bounded control effort[1]. Our model in (2.1) can readily account for multiple agents via state augmentation and the addition of multiple control inputs. In the case of k agents, our dynamics will take the form:

$$\dot{x} = f(x, u_1, \ldots, u_k) \tag{2.2}$$

where the augmented state is $x \in \mathbb{R}^{\sum_{i=1}^{k} n_i}$, where n_i is the dimension of the ith agent's state space.

[1]In general, the dynamics can be dependent on time. However, in this dissertation the dynamics f will be time-invariant, allowing us to drop the explicit time dependence

When studying (2.1), a question naturally appears: can we always solve this ODE? In other words, is the concept of a "state trajectory" (which would be the solution to this ODE) well-defined? It turns out that if we ensure that our flow field, $f : \mathbb{R}^n \times \mathcal{U} \to \mathbb{R}^n$, abides by some key properties, then the answer to this question will be yes! Specifically, it can be shown that if the flow field is uniformly continuous in its arguments, and Lipschitz continuous in x for all control inputs $u \in \mathcal{U}$, then there exists a unique solution to our system dynamics for a given control signal $\mathbf{u}(\cdot) : [t_0, \infty) \to \mathcal{U}$. Mathematically, these two conditions are met by the resultant state trajectory:

$$\dot{\mathbf{x}}(t) = f(\mathbf{x}(t), \mathbf{u}(t)) \quad a.e. \quad t \geq t_0 \tag{2.3}$$

$$\mathbf{x}(0) = x_0 \tag{2.4}$$

Aside on Notation. *State and control inputs are functions of time, thus the most precise notation would be $x(t) \in \mathbb{R}^n, u(t) \in \mathcal{U}$. However, when the dependence on time is clear, we will often drop the explicit dependence on t to simplify notation.*

A state trajectory in continuous-time will be denoted as $\mathbf{x}(\cdot) \in \mathbb{X}_t^T$, and a control trajectory (or signal) $\mathbf{u}(\cdot) \in \mathbb{U}_t^T$, indicating that these trajectories go from time t to T. To denote a specific state or control along a trajectory, we index it in time: $\mathbf{x}(t), \mathbf{u}(t)$. If we want to clearly denote a state trajectory which begins from a particular initial state x at time t and under a particular control signal $\mathbf{u} \in \mathbb{U}_t^T$ we will write $\mathbf{x}_{x,t}^{\mathbf{u}}(\cdot) \in \mathbb{X}_t^T$

2.1.2 Discrete-Time Dynamical Systems

Thus far, we discussed continuous-time dynamical systems. However, discrete-time systems are widely used in the AI, robotics, and reinforcement learning communities. Let the discrete-time dynamics be represented by:

$$x^{t+1} = \tilde{f}(x^t, u^t), \quad t \in \mathbb{N}. \tag{2.5}$$

Here, $x^t \in \mathbb{R}^n$ and $u^t \in \mathcal{U}$ are the state and control inputs indexed at the discrete time t, and $\tilde{f}$ is map from the current state and control to next state. Often times, we may have access to a continuous-time dynamics model, but need to discretize it to perform computations on a computer. A common method for discrete-time approximations is a first-order Euler approximation:

$$\tilde{f}(x^t, u^t) := x^t + \delta t f(x^t, u^t) \tag{2.6}$$

where δt is a time-step. Higher order discretizations may also be used to improve the accuracy of the approximation at the expense of computation speed.

Aside on Notation. *When we want to denote a discrete-time trajectory (also known as a sequence) from time $t = 0$ to T, we will use $\mathbf{x} := [x^0, \dots x^T]$ or $x^{0:T}$ to denote the state trajectory and $\mathbf{u} := [u^0, \dots u^T]$ or $u^{0:T}$ to denote the control trajectory. Sometimes we will want to denote a sequence of states and controls. We use the notation $\xi := \{(x^0, u^0), \dots, (x^{T-1}, u^{T-1})\}$* for the state-action trajectory.

2.1.3 Stochastic Discrete-Time Dynamical Systems

So far, we have discussed deterministic dynamics models. However, stochastic dynamics models are core to many robotics and human-robot interaction problems. In this subsection, we restrict ourselves to discussing the discrete-time stochastic control systems, also known as Markov Decision Processes (MDPs). This is a popular modelling tool in the AI, reinforcement learning, and human modelling community. Modelling the dynamics as *stochastic* means that executing a control u^t from a state x^t does not always result in the same next state x^{t+1}. Concretely, let $X^{t+1} \subseteq \mathbb{R}^n$ be a random variable representing the next state. Given a current state x^t and a current control input u^t, the evolution of the dynamical system can be described by the probability distribution:

$$P(X^{t+1} \mid x^t, u^t). \tag{2.7}$$

In the MDP and Reinforcement Learning communities, (2.7) is often called the *transition function*. Note that this model is *Markovian*, meaning that the next state depends only on the current state and not on the full state history. This enables us to easily reason about state probabilities at any time in the future. Just like in 2.1.1, if we are given a discrete-time control sequence, $u^{0:t}$, we may want to know the likelihood of any future state at time $t \in \{0, 1, \dots T\}$, i.e., we want to obtain a distribution over X^{t+1}. Using our stochastic dynamics, we can write:

$$P(X^{t+1} = x^{t+1} \mid x^0, u^{0:t}) = \int_{\mathbb{R}^n} \dots \int_{\mathbb{R}^n} \prod_{\tau=0}^{t} P(x^{\tau+1} \mid x^\tau, u^\tau) dx^1 \dots dx^t \tag{2.8}$$

which quantifies the probability of the system ending up at a particular state x at future time $t + 1$.

2.2 From Optimal Control to Single-Agent Safety

In the previous section, we understood how to mathematically model a dynamical system that evolves as a function of its control inputs. But how should an autonomous system *choose* its control inputs? What even constitutes a “good” control input?

Optimal control addresses exactly this problem: automatically obtaining control inputs which are optimal under a given decision-making objective. For example, figuring out the acceleration profile *(control input)* which smoothly but efficiently moves a car through an intersection *(decision-making objective)* is an optimal control problem. In this section, we will formalize the optimal decision-making problem for a single agent (like a robot), discuss variants of posing and solving optimal control problems, and show how a special class of optimal control problems allow us to synthesize provably-safe controllers for robots acting in isolation.

2.2.1 Formalizing the Optimal Control Problem

Consider a deterministic, continuous-time dynamical system like that in (2.1), evolving over the time interval $[0, T]$. Mathematically, the optimal control problem consists of minimizing a cost function (also called *objective function*, or *utility function*) subject to the dynamics and control constraints. Let the cumulative cost we seek to minimize over the time interval $[t, T]$ where $0 \leq t \leq T$ be:

$$J(x, \mathbf{u}(\cdot), t) := \int_t^T c(x(\tau), u(\tau))d\tau + \ell(x(T)) \tag{2.9}$$

where $c(\cdot, \cdot)$ is the instantaneous cost of being at state x and applying control u (for example, how much fuel is being used), and $\ell(\cdot)$ is the terminal cost of ending up at a state x at time T (for example, distance to goal). The cumulative cost function $J : \mathbb{R}^n \times (\bigcup_{t\in[0,T]} \mathbb{U}_t^T \times t) \to \mathbb{R}$ takes as input a starting state, a control signal, and also the starting time, t, and outputs the cost accumulated over the time horizon $[t, T]$.

We seek to find a control signal $\mathbf{u}(\cdot) \in \mathbb{U}_t^T$ which minimizes this cumulative cost:

$$\begin{aligned} V(x, t) := \inf_{\mathbf{u}(\cdot)\in\mathbb{U}_t^T} \quad & J(x, \mathbf{u}(\cdot), t) \\ \text{s.t.} \quad & \dot{x}(\tau) = f(x(\tau), u(\tau)), \quad \forall \tau \in [t, T] \\ & u(\tau) \in \mathcal{U} \end{aligned} \tag{2.10}$$

while ensuring that the system evolves according to the dynamics and respecting control bounds, $\mathcal{U}$. We can keep track of the best possible cost we can achieve over this time interval from t to T through the *value function*, $V(x, t)$.

If we had a discrete-time system like in (2.5), the optimal control problem would look similar, barring a few changes. The cost function would now be:

$$J^t(x, \mathbf{u}) := \sum_{\tau=t}^{T-1} c(x^\tau, u^\tau) + \ell(x^T) \tag{2.11}$$

and the optimal control problem is

$$\begin{aligned} V^t(x) := \min_{\mathbf{u}} \quad & J^t(x, \mathbf{u}) \\ \text{s.t.} \quad & x^{\tau+1} = \tilde{f}(x^\tau, u^\tau), \quad \forall \tau \in \{t, \dots, T-1\} \\ & u^\tau \in \mathcal{U} \end{aligned} \tag{2.12}$$

where $V^t(x)$ is the value of the discrete-time optimal control problem.

For the rest of this section, we will focus on the continuous-time optimal control problem, but highlight key differences between that and the discrete-time formulation when relevant.

2.2.2 Solving the Optimal Control Problem

Now that we have posed our desired optimal control problem in (2.10), how do we actually solve it? The optimization and control communities have established a huge suite of methods for solving optimal control problems, but the two most foundational approaches are the calculus of variations and dynamic programming.

Calculus of variations. The key idea behind this approach is to carefully transform the constrained optimization problem into an unconstrained one and then pose a set of necessary conditions that an optimal control trajectory must satisfy. Intuitively, this process begins by moving each constraint (e.g., the dynamics) into the objective function and associating each with a *Lagrange multiplier*. Next, we need to know the necessary conditions that any optimal control trajectory and the Lagrange multipliers must satisfy. Luckily for us, *Pontryagin's maximum principle* precisely states this series of first-order necessary condition that any optimal control trajectory and the Lagrange multipliers must satisfy [168]. The big advantage of the calculus of variations approach is that once we obtain this set of necessary conditions, we can leverage a myriad of (often very efficient) computational tools developed by the optimization community to solve for the control trajectory. Unfortunately, the solution obtained by this method is not guaranteed to be globally optimal (most of the time it is only locally optimal) unless the optimal control problem is convex in the decision variable and the duality gap is zero.

Dynamic programming. The key idea of this approach is to reason recursively about the optimal control problem backwards in time. Unlike the calculus of variations approach, this approach guarantees a globally optimal solution at the expense of significantly higher computational costs. The reason for this globally optimal solution is the construction of the Hamilton-Jacobi-Bellman (HJB) partial differential equation (PDE). In discrete-time, this equation is called the *Bellman equation* or *Bellman backup*, and is popular in both the control, AI, and reinforcement learning communities. These equations capture the necessary and sufficient conditions that the optimal control problem's value function must satisfy. Solving the HJB-PDE (or the Bellman equation) backwards in time ensures that the solution over the entire control trajectory is globally optimal.

2.2.3 Dynamic Programming: From Continuous to Discrete Time

From guaranteeing safe robot operation to modelling human behavior, the principle of dynamic programming is foundational to the majority of approaches discussed in this dissertation. The idea of dynamic programming, introduced by Richard Bellman in the 1950s, solves optimal control problems by leveraging the *principle of optimality*:

> **Principle of Optimality [30].** An optimal policy has the property that whatever the initial state and initial decisions are, the remaining decisions must constitute and optimal policy with regard to the state resulting from the first decision.

In other words, assume you are given an optimal sequence of controls and the corresponding optimal state trajectory. If you start at *any* state along that optimal state trajectory, then the remaining sub-trajectory is also optimal. The implication of this is that we can solve optimal control problems by combining optimal solutions of smaller sub-problems.

2.2.4 The Hamilton-Jacobi-Bellman PDE

Let's understand the principle of optimality in the context of continuous-time optimal control problems. Recall the *value function* we defined earlier starting from state x at time t:

$$V(x,t) = \inf_{\mathbf{u}(\cdot)\in\mathbb{U}_t^T} \int_t^T c(x(\tau),u(\tau))d\tau + \ell(x(T)). \tag{2.13}$$

The principle of optimality tells us that we can divide the computation of this value function in time by creating two sub-problems: solving for the optimal control signal for $\tau \in [t, t+\delta)$ and for $\tau \in [t+\delta, T]$:

$$V(x,t) = \inf_{\mathbf{u}(\cdot)\in\mathbb{U}_t^{t+\delta}} \int_t^{t+\delta} c(x(\tau),u(\tau))d\tau + \inf_{\mathbf{u}(\cdot)\in\mathbb{U}_{t+\delta}^T} \int_{t+\delta}^T c(x(\tau),u(\tau))d\tau + \ell(x(T)). \tag{2.14}$$

Note that the starting state within the integral $\int_{t+\delta}^T$ must be $x(t+\delta)$, the dynamically-feasible state we reach at time $t+\delta$ after applying the control signal $\mathbf{u}(\cdot)$ over the time horizon $[t, t+\delta]$. We can go one step further and insert the value function *at the future state $x(t+\delta)$ and future time $t+\delta$* into the right-hand side of (2.14) because of the definition of the value function in (2.13). We know that $V(x(t+\delta), t+\delta)$ captures the best we could ever hope to do starting from $x(t+\delta)$ and evolving along that future sub-trajectory! In other words, the value function is the solution to the second sub-problem we posed. Inserting it into (2.14), we obtain:

$$V(x,t) = \inf_{\mathbf{u}(\cdot)\in\mathbb{U}_t^{t+\delta}} \int_t^{t+\delta} c(x(\tau),u(\tau))d\tau + V(x(t+\delta), t+\delta). \tag{2.15}$$

Last but not least, at the final time $\tau = T$, the value at the final state and time is simply equal to the terminal cost, $V(x,T) = \ell(x(T))$.

But why does reformulating the value function to look like (2.15) even matter? Well, now that we have reduced the problem to only optimizing our control signal over the smaller time-horizon of $[t, t+\delta]$, we can more easily think about what happens when $\delta \longrightarrow 0$. In other words, how does the value at the current state and time change when we make small changes in our decisions?

For now, assume that V is everywhere differentiable. Let's go through an informal derivation of how our value changes when $\delta > 0$ but is very small. In this scenario, the value function from (2.15) can be written as

$$V(x,t) \approx \min_{u\in\mathcal{U}} \Big\{ c(x,u)\delta + V(x(t+\delta), t+\delta) \Big\}. \tag{2.16}$$

Expanding $V(x(t+\delta), t+\delta)$ via the Taylor's series expansion around the current state and time, $(x(t), t)$, we get:

$$V(x(t+\delta), t+\delta) \approx V(x(t), t) + \frac{\partial V(x(t), t)}{\partial t}(t+\delta-t) + \nabla_x V(x(t), t) \cdot (x(t+\delta) - x(t)) + \text{h.o.t}, \tag{2.17}$$

where the change in state can be approximated via the dynamics: $x(t+\delta) - x(t) \approx f(x, u)\delta$. Ignoring the higher order terms, we obtain:

$$V(x(t+\delta), t+\delta) \approx V(x(t), t) + \frac{\partial V(x(t), t)}{\partial t}\delta + \nabla_x V(x(t), t) \cdot f(x(t), u(t))\delta. \tag{2.18}$$

Plugging this approximation to $V(x(t+\delta), t+\delta)$ into our original equation and moving the terms that do not depend on control outside of the minimization, we obtain:

$$V(x, t) \approx \min_{u \in \mathcal{U}} \left\{ c(x, u)\delta + V(x, t) + \frac{\partial V(x, t)}{\partial t}\delta + \nabla_x V(x, t) \cdot f(x, u)\delta \right\} \tag{2.19}$$

$$= V(x, t) + \frac{\partial V(x, t)}{\partial t}\delta + \min_{u \in \mathcal{U}} \left\{ c(x, u)\delta + \nabla_x V(x, t) \cdot f(x, u)\delta \right\}. \tag{2.20}$$

Cancelling out the redundant terms on both sides, we obtain:

$$0 = \frac{\partial V(x, t)}{\partial t}\delta + \min_{u \in \mathcal{U}} \left\{ c(x, u)\delta + \nabla_x V(x, t) \cdot f(x, u)\delta \right\} \tag{2.21}$$

$$\approx \frac{\partial V(x, t)}{\partial t} + \min_{u \in \mathcal{U}} \left\{ c(x, u) + \nabla_x V(x, t) \cdot f(x, u) \right\} \qquad \text{(divide by } \delta > 0\text{)} \tag{2.22}$$

With this, we have obtained the Hamilton-Jacobi-Bellman partial differential equation:

$$\frac{\partial V(x, t)}{\partial t} + \min_{u \in \mathcal{U}} \left\{ c(x, u) + \nabla_x V(x, t) \cdot f(x, u) \right\} = 0 \tag{2.23}$$

$$V(x, T) = \ell(x). \tag{2.24}$$

with the terminal condition $V(x, T) = \ell(x)$. We can solve this final-value PDE backwards in time starting from the terminal time T to obtain the value function $V(x, t)$ for any state and time. Then, we can obtain the optimal control at any given state and time via

$$u^*(x, t) = \arg\min_{u \in \mathcal{U}} \left\{ c(x, u) + \nabla_x V(x, t) \cdot f(x, u) \right\}. \tag{2.25}$$

Just like when we were studying the dynamics ODE in 2.1.1 and asking "does a solution to this ODE even exist?", we can ask the same question of the HJB-PDE. Traditionally, the solution to this PDE only existed if we assumed that the value function was differentiable everywhere (we assumed this earlier in our intuitive derivation). However, this is too stringent of a requirement for the majority of optimal control problems we will seek to solve; for example, our system trajectories themselves only have to satisfy the dynamics ODE *almost everywhere* in time. Fortunately, mathematicians Crandall and Lions [57] introduced a novel solution concept for the HJB-PDE called the *viscosity solution* which states that at all non-differentiable points, V must satisfy a relaxed condition which bounds the value function from above and below with continuously differentiable functions.

> ***Aside on solving the HJB-PDE.*** *Solving the HJB-PDE for the optimal value function (and corresponding optimal control) is not trivial. In certain conditions, like in the event that the running cost $c(\cdot,\cdot)$ and the terminal cost $\ell(\cdot)$ are quadratic in state, and the dynamics are linear (i.e., $\dot{x} = Ax(t) + Bu(t)$), we can obtain the value function and optimal control in closed-form. This enables us to efficiently compute a very high-fidelity value function, since we do not have to discretize state, time, or control. However, for most dynamical systems and cost functions, we will need to numerically integrate the HJB-PDE. One approach is to discretize time and state, inducing a tabular or grid-based representation of V, causing the computational complexity of solving the HJB-PDE to scale exponentially in the state dimensionality. This is referred to as the* curse of dimensionality *and is the topic of extensive study (see [25] for an overview of recent advances). An alternative is to approximate the value function directly. There is a wealth of such techniques, including neural PDE solvers [163, 63, 181, 105, 24], sum-of-squares (SOS) techniques [131], or zonotope approximations [227, 82, 119]. For a taxonomy of how the value function is computed under different methods and conditions, see Fig. 2.1.*

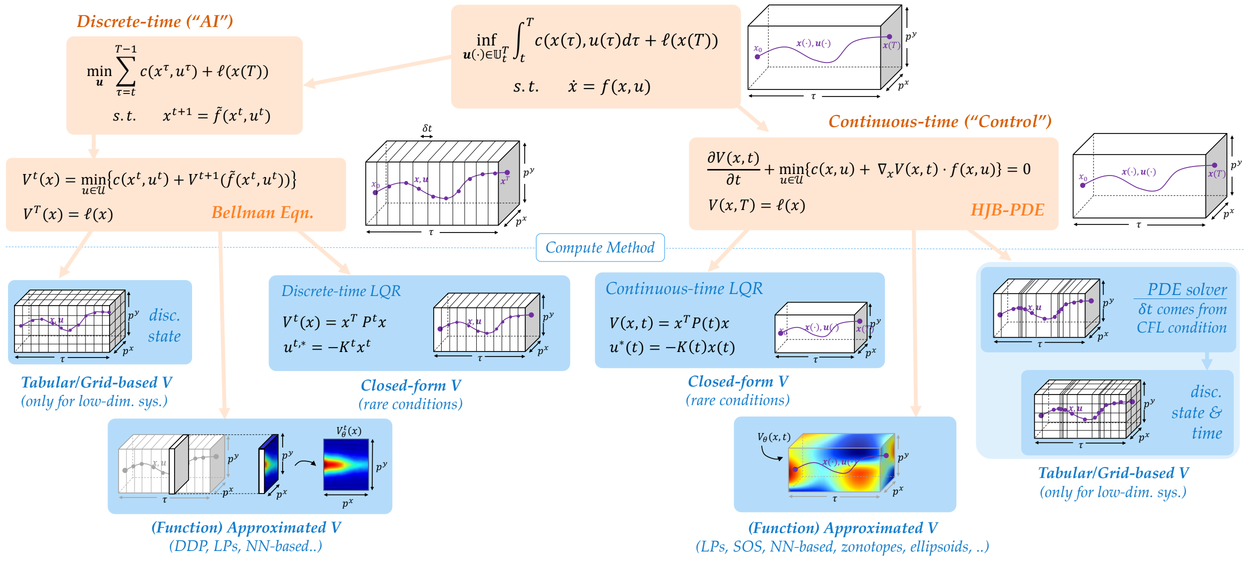

Discrete-time ("AI")
$\min_{u} \sum_{\tau=t}^{T-1} c(x^\tau, u^\tau) + \ell(x(T))$
$s.t. \quad x^{t+1} = \tilde{f}(x^t, u^t)$
$\inf_{u(\cdot)\in\mathbb{U}_t^T} \int_t^T c(x(\tau), u(\tau)d\tau + \ell(x(T))$
$s.t. \quad \dot{x} = f(x,u)$
x_0
$x(\cdot), u(\cdot)$
$x(T)$
p^y
p^x
τ
Continuous-time ("Control")
$V^t(x) = \min_{u\in\mathcal{U}}\{c(x^t,u^t) + V^{t+1}(\tilde{f}(x^t,u^t))\}$
$V^T(x) = \ell(x)$
Bellman Eqn.
δt
x, u
x^T
$\frac{\partial V(x,t)}{\partial t} + \min_{u\in\mathcal{U}}\{c(x,u) + \nabla_x V(x,t) \cdot f(x,u)\} = 0$
$V(x,T) = \ell(x)$
HJB-PDE
Compute Method
disc. state
Tabular/Grid-based V
(only for low-dim. sys.)
Discrete-time LQR
$V^t(x) = x^T P^t x$
$u^{t,*} = -K^t x^t$
Closed-form V
(rare conditions)
Continuous-time LQR
$V(x,t) = x^T P(t) x$
$u^*(t) = -K(t)x(t)$
Closed-form V
(rare conditions)
PDE solver
δt comes from CFL condition
disc. state & time
Tabular/Grid-based V
(only for low-dim. sys.)
$V_\theta^t(x)$
(Function) Approximated V
(DDP, LPs, NN-based..)
$V_\theta(x,t)$
(Function) Approximated V
(LPs, SOS, NN-based, zonotopes, ellipsoids, ..)

Figure 2.1: Taxonomy of optimal control problems in discrete and continuous time, as well their value function computation and representation.

2.2.5 The Bellman Equation

So far we have seen a dynamic programming solution to the continuous-time optimal control problem. However, discrete-time optimal control problems are very popular in the AI, robotics, and reinforcement learning communities.

Recall the value function of the discrete-time optimal control problem from (2.12):

$$V^t(x) := \min_{\mathbf{u}} \sum_{\tau=t}^{T-1} c(x^\tau, u^\tau) + \ell(x^T). \tag{2.26}$$

We know that at the final time, the value will be $V^T(x) = \ell(x^T)$. Then, starting at some time $0 \leq t < T$, we can apply the principle of optimality and the definition of the value function, splitting up the optimization problem into two subproblems:

$$\begin{aligned} V^t(x) &= \min_{u^t \in \mathcal{U}} \left\{ c(x^t, u^t) + \min_{\mathbf{u}:=[u^{t+1} \ldots u^{T-1}]} \sum_{\tau=t+1}^{T-1} c(x^\tau, u^\tau) + \ell(x^T) \right\} && (2.27) \\ &= \min_{u^t \in \mathcal{U}} \left\{ c(x^t, u^t) + V^{t+1}(x^{t+1}) \right\}. && (2.28) \end{aligned}$$

With this, we have obtained the *Bellman equation* [30]:

$$\begin{aligned} V^t(x) &= \min_{u^t \in \mathcal{U}} \left\{ c(x^t, u^t) + V^{t+1}(x^{t+1}) \right\} && (2.29) \\ V^T(x) &= \ell(x). && (2.30) \end{aligned}$$

The HJB-PDE in (2.24) can be thought of as the continuous-time equivalent of the discrete-time Bellman equation. However, unlike in continuous-time, dynamic programming in discrete time yields a value function "backup" instead of a PDE. See Fig. 2.1 for a comparison of the continuous and discrete-time solutions to deterministic optimal control problems, and how they are computed under different conditions and methods.

The Bellman equation is also very popular for solving optimal control problems with *stochastic* dynamics. At its core, this optimal control problem seeks to minimize the *expected* cost:

$$V^t(x) := \min_{\mathbf{u}} \quad \mathbb{E}_{\mathbf{x} \sim P(\mathbf{x} | x^t, \mathbf{u})} \left[\sum_{\tau=t}^{T-1} c(x^\tau, u^\tau) + \ell(x^T) \right] \tag{2.31}$$

where $P(\mathbf{x} \mid x^t, \mathbf{u})$ is the probability of the system entering a sequence of states $\mathbf{x}$ starting from state x^t and executing the control sequence $\mathbf{u}$.

The corresponding discrete-time Bellman equation is:

$$V^t(x) = \min_{u^t \in \mathcal{U}} \left\{ c(x^t, u^t) + \int_{\mathbb{R}^n} P(x^{t+1} \mid x^t, u^t) V^{t+1}(x^{t+1}) dx^{t+1} \right\}. \tag{2.32}$$

See [31] for a more in-depth treatment of the discrete-time value function, continuous-time value function, and their connections.

2.2.6 Hamilton-Jacobi Reachability & Single-Agent Safety

We now have a handle on how to solve general optimal control problems for a single agent, like a robot. However, right now the optimal control problems we have posed minimize cost and constrain that the system evolves via the dynamics. What if we wanted to guarantee that the system abides by some state constraints? For example, what if we want to come up with an optimal control which *guarantees* that our robot never hits an obstacle? Or, an optimal control trajectory which guarantees that the robot reaches its goal? Answering these questions and synthesizing the corresponding control trajectories are core to *safety analysis*. It is important to note that these safety specifications are treated as *hard constraints* in the optimization, instead of being part of the objective which can be traded off (e.g., like efficiency, fuel-consumption, etc.).

While there are a myriad of approaches for verifying the safety and performance of dynamical systems, this dissertation is rooted in Hamilton-Jacobi (HJ) reachability analysis which is built upon the optimal control and dynamic programming tools we have discussed thus far. In fact, in this section we will discuss how these safety questions (about guaranteeing obstacle avoidance or goal-reaching) can be posed and solved as a special type of optimal control problem.

Reach Problems (i.e., goal satisfaction)

Let's start our journey into reachability by studying *reach* problems. This class of problems are concerned with determining if there exists a control signal that can drive a system to a desired state. For example, imagine you are an autonomous vehicle (AV) designer and you are tasked with developing how an AV should park itself into a user's garage. If the AV starts in the driveway, does there exist a control signal that guides it into the garage? What if the autonomous vehicle started at an odd angle and then tried to maneuver into the garage? Does there exist a control signal that can still steer it into the garage successfully? Hamilon-Jacobi reachability gives us not only a binary answer to these questions, but it also produces the optimal control signal in the event that the binary answer to these questions is "yes, we can control our system to the desired goal".

Backwards Reachable Set. The *backwards reachable set (BRS)* of a dynamical system is the set of states from which if the system begins, then it can reach a given target set (e.g., the robot's goal) at the final time. Mathematically, let's define the target set to be $\mathcal{L} \subseteq \mathbb{R}^n$. The BRS is the set of states $x \in \mathbb{R}^n$ from which the system can be driven into $\mathcal{L}$ *exactly at the end of the time horizon*.

$$\mathcal{V}_{BRS}(t) = \{x : \exists \mathbf{u}(\cdot) \in \mathbb{U}_t^T, \mathbf{x}_{x,t}^{\mathbf{u}}(T) \in \mathcal{L}\}. \tag{2.33}$$

But how do we actually obtain this set? It turns out that we can transform this goal satisfaction problem into an optimal control problem via the *level set method*. Intuitively, we will design a specific type of optimal control objective which captures *how close our system is to satisfying our goal-reaching constraint*. By measuring how close the system is to reaching the goal, our optimal control problem can find whether there exists a control signal which minimizes the distance between the system starting from some initial state and the goal; in other words, if our system can achieve goal satisfaction.

Specifically, we define a Lipschitz function $\ell(x)$ such that our target set $\mathcal{L}$ is equal to the sub-zero level set of this function: $x \in \mathcal{L} \iff \ell(x) \leq 0$. We can always find such a function ℓ to encode the target set by simply choosing the signed distance to the set boundary. Now we can modify our optimal control objective to be:

$$J(x, \mathbf{u}(\cdot), t) = \ell(\mathbf{x}_{x,t}^{\mathbf{u}}(T)) \tag{2.34}$$

where the controller only cares about how close the final state is to the target (as measured by $\ell(\cdot)$). The overall optimal control problem is now:

$$\begin{aligned} V(x,t) := \inf_{\mathbf{u}(\cdot)\in\mathbb{U}_t^T} \quad & J(x, \mathbf{u}(\cdot), t) \equiv \ell(\mathbf{x}_{x,t}^{\mathbf{u}}(T)) \\ \text{s.t.} \quad & \dot{x}(\tau) = f(x(\tau), u(\tau)), \quad \forall \tau \in [t, T] \\ & u(\tau) \in \mathcal{U} \end{aligned} \tag{2.35}$$

It is now clear to see how this is just a modified version of the original optimal control objective from (2.9) where $c(x(t), u(t)) = 0$ and the terminal cost $\ell(\cdot)$ encodes the distance to our target set $\mathcal{L}$.

Now that we are back in optimal control land, we already have the tools necessary to solve this problem! In fact, we can use the machinery from 2.2.3 and solve the HJB-PDE to obtain the value function for this optimal control problem. Since any control trajectories that are able to drive the system into the target set will be assigned a negative value by ℓ, we know that all states whose value is ≤ 0 must be part of our BRS. Consequently, the BRS can be obtained from the value function via:

$$\mathcal{V}_{BRS}(t) = \{x : V(x,t) \leq 0\}. \tag{2.36}$$

What if we want to obtain the optimal control trajectory which takes us from a start state $x \in \mathcal{V}_{BRS}(t)$ to the target set? With access to the value function, this is also straightforward:

$$u^*(x,t) = \arg\min_{u\in\mathcal{U}} \nabla_x V(x,t) \cdot f(x,u) \tag{2.37}$$

Backwards Reachable Tube. We can also pose a slightly different, but related question: what are the set of states $x \in \mathbb{R}^n$ from which the system can be driven into $\mathcal{L}$ *at any time during the time horizon*? In the reachability literature, this set is called a *backwards reachable tube (BRT)*. Recall that the BRS demands that the system trajectory enters the target set $\mathcal{L}$ *exactly* at the final time. This means that if a system trajectory enters the goal and then leaves it before the final time, then it has failed to satisfy the constraint. The nice aspect of BRTs is that trajectories which enter the target set $\mathcal{L}$ at some early time and then leave it are still considered to have satisfied the goal-reaching constraint. Mathematically, the set of states which satisfy this property are:

$$\mathcal{V}_{BRT}(t) = \{x : \exists \mathbf{u}(\cdot) \in \mathbb{U}_t^T, \exists \tau \in [t,T], \mathbf{x}_{x,t}^{\mathbf{u}}(\tau) \in \mathcal{L}\}. \tag{2.38}$$

To convert this into our optimal control framework, we need to add an extra minimization over time to "keep track" of the event that a trajectory has entered into the target set at some time. If a trajectory does enter the target set (and therefore is assigned a negative value at that state and time by ℓ), then the minimization will preserve that negative value for that system trajectory over the entire optimization horizon even if the trajectory leaves the target and starts being assigned positive values by ℓ. Mathematically, this optimization problem is:

$$\begin{aligned} V(x,t) := \inf_{\mathbf{u}(\cdot)\in\mathbb{U}_t^T} \min_{\tau\in[t,T]} \quad & \ell(\mathbf{x}_{x,t}^{\mathbf{u}}(\tau)) \\ \text{s.t.} \quad & \dot{x}(\tau) = f(x(\tau), u(\tau)), \quad \forall \tau \in [t,T] \\ & u(\tau) \in \mathcal{U} \end{aligned} \tag{2.39}$$

Note that now our cost function no longer satisfies the typical form from (2.10) since the optimal control signal is finding the minimum of a function over the time horizon (instead of the integral). Fortunately, we can still use the principle of dynamic programming to compute the optimal value function. After applying the dynamic programming principle, we can obtain the following final-value *Hamilton-Jacobi Variational Inequality (HJB-VI)*:

$$\min\left\{\ell(x) - V(x,t), \frac{\partial V(x,t)}{\partial t} + \min_{u\in\mathcal{U}}\left\{\nabla_x V(x,t)\cdot f(x,u)\right\}\right\} = 0 \tag{2.40}$$

$$V(x,T) = \ell(x). \tag{2.41}$$

Finally, the BRT can be extracted from the value function, which is the viscosity solution to (2.41):

$$\mathcal{V}_{BRT}(t) = \{x : V(x,t) \leq 0\} \tag{2.42}$$

Forward Reachability. The final variant of a reach problem that we may be interested in is computing something called a *forward reachable set (FRS)* or *forward reachable tube (FRT)*. The FRS captures the set of all states that a system can reach from a set of initial states at *exactly* time T. The FRT captures the set of all states that a system can reach from a set of initial states at *within* the time horizon $[t,T]$. These are formally defined as:

$$\mathcal{V}_{FRS}(t) := \{y : \exists \mathbf{u}(\cdot) \in \mathbb{U}_t^T, x_0 \in \mathcal{L}, \mathbf{x}_{x_0,t}^{\mathbf{u}}(T) = y\} \tag{2.43}$$

$$\mathcal{V}_{FRT}(t) := \{y : \exists \mathbf{u}(\cdot) \in \mathbb{U}_t^T, x_0 \in \mathcal{L}, \exists \tau \in [t,T], \mathbf{x}_{x_0,t}^{\mathbf{u}}(\tau) = y\}. \tag{2.44}$$

While in the backwards reachability setting we had $\mathcal{L}$ represent a desired set of states we wanted to *reach*, here, $\mathcal{L}$ represents the set of initial states our system *starts* from.

The FRS can be solved for in a similar fashion as the BRS, except we need to solve an initial-value PDE instead of a final-value one:

$$\frac{\partial V(x,t)}{\partial t} + \min_{u \in \mathcal{U}} \left\{ \nabla_x V(x,t) \cdot f(x,u) \right\} = 0 \tag{2.45}$$

$$V(x,0) = \ell(x). \tag{2.46}$$

Note that the only change is that the value function is initialized at the *initial* time zero, and is set to encode the initial states our system starts from. After solving this initial-value HJB-PDE for the value function, we can obtain the FRS as the subzero level set of the value function:

$$\mathcal{V}_{FRS}(t) = \{x : V(x,t) \leq 0\}. \tag{2.47}$$

The FRT can be computed in a nearly identical fashion, except we solve the *initial-value* version of the HJB-VI from (2.41). Obtaining the value function from this problem, we can once again obtain the FRT, $\mathcal{V}_{FRT}$, via the subzero level set of the value function.

Avoid Problems (i.e., safety satisfaction)

Conceptually, we can think of avoid problems as the inverse of reach problems. Here, we want to determine if there exists a control signal which can always steer the system *away* from an *undesirable* set of states. Let's go back to our AV designer example. Instead of parking into the garage, you are now faced with designing how an AV should exit the garage without hitting the trashcans nearby. If the AV starts really close to the trashcans, can it still avoid colliding into them? Or is collision inevitable? In the following subsections, we will formalize these questions (and their answers) mathematically. In fact, we will discover that not much mathematical machinery needs to change from the reach problems to solve avoid problems!

Backwards Avoid Sets and Tubes. The *backwards avoid set*, also known as the *avoid backwards reachable set (BRS)*, of a dynamical system is very similar to the backwards reachable set from earlier. However, here, the avoid BRS is the set of states from which if the system begins, it is doomed to enter into the undesirable set of states. Mathematically, we will use the same notion of a target set to be $\mathcal{L} \subseteq \mathbb{R}^n$, however this is now capturing the set of states we seek to *avoid*. The avoid BRS is the set of states $x \in \mathbb{R}^n$ from which the system is doomed to enter $\mathcal{L}$ *at exactly at the end of the time horizon*.

$$\mathcal{V}_{BRS}(t) = \{x : \forall \mathbf{u}(\cdot) \in \mathbb{U}_t^T, \mathbf{x}_{x,t}^{\mathbf{u}}(T) \in \mathcal{L}\}. \tag{2.48}$$

Note that the only difference between this equation and the previous one in (2.33) is the change from $\exists \mathbf{u}(\cdot)$ to $\forall \mathbf{u}(\cdot)$. The change to "for all" control signals captures how no matter what the system does control-wise, it is doomed to enter the undesirable states $\mathcal{L}$ in T seconds. We can also make the same change to the definition of our *backwards avoid tube*, or *avoid backwards reachable tube (BRT)*. Here the avoid BRT captures the set of initial states from which if the system begins, it is doomed to enter the undesirable states $\mathcal{L}$ at any time during $\tau \in [t, T]$:

$$\mathcal{V}_{BRT}(t) = \{x : \forall \mathbf{u}(\cdot) \in \mathbb{U}_t^T, \exists \tau \in [t, T], \mathbf{x}_{x,t}^{\mathbf{u}}(\tau) \in \mathcal{L}\}. \tag{2.49}$$

Converting this into our optimal control framework, we only need to make one small change to (2.35) and (2.39): changing the $\inf_{\mathbf{u}(\cdot)}$ to $\sup_{\mathbf{u}(\cdot)}$. What's the intuition behind this change? Recall that we were using the level set method, where we constructed a function ℓ which measures how close our system is to reaching a set of states $\mathcal{L}$. If the system is inside $\mathcal{L}$, it obtains a negative value, if it is on the boundary it obtains zero value, and if it is outside, it obtains a positive value. Before, when the system was trying to *reach* $\mathcal{L}$, the control objective was to *minimize* the cost ℓ. However, now that $\mathcal{L}$ encodes states that we want to *avoid*, the control is trying to *maximize* the objective ℓ, thereby staying outside of the undesirable region. However, if all control signals achieve a negative cost, then we know that entering into the undesirable region is unavoidable.

To obtain the avoid BRS, we solve the HJB-PDE from (2.24); for the avoid BRT, we solve the HJB-VI (2.41). The only change in the formulation is the replacement of the $\min_u$ to $\max_u$. For example, the HJB-PDE which will help us compute the avoid BRS is:

$$\frac{\partial V(x,t)}{\partial t} + \max_{u \in \mathcal{U}} \left\{ c(x,u) + \nabla_x V(x,t) \cdot f(x,u) \right\} = 0 \tag{2.50}$$

$$V(x,T) = \ell(x). \tag{2.51}$$

To obtain the avoid sets themselves, $\mathcal{V}_{BRS}$ and $\mathcal{V}_{BRT}$, we can take the value function computed by solving the corresponding PDEs, and obtain the subzero level set of the value functions.

How can we use these avoid sets in practice? Let's consider the avoid BRT. Recall that it captures the set of states from which, if our system starts, it is doomed to enter into the undesirable set of states at some time $\tau \in [t, T]$. In addition to this, the complement of this set, $\mathcal{V}_{BRT}^{\mathsf{c}}$, is the set of guaranteed *safe* initial conditions for our system. In other words, as long as the system starts *outside* of the avoid BRT, then a controller exists to steer it away from the undesirable states. Specifically, for any $\tau \in [t, T]$, this safe controller is:

$$u^*(x,t) = \arg\max_{u \in \mathcal{U}} \nabla_x V(x,t) \cdot f(x,u) \tag{2.52}$$

where $V(\cdot,\cdot)$ is the value function computed by solving the avoid BRT's HJB-VI. Intuitively, using this optimal control will have the system "try its hardest" to avoid entering the unsafe states.

For a summary of these many variants of reachability problems, their corresponding optimal control problems, and their solutions, please refer to Fig. 2.2.

2.2.7 Practical Notes

Least-restrictive, safety-preserving controller. So far, we have been working under the assumption that the robot needs to use the safety-preserving control law (2.52) at all

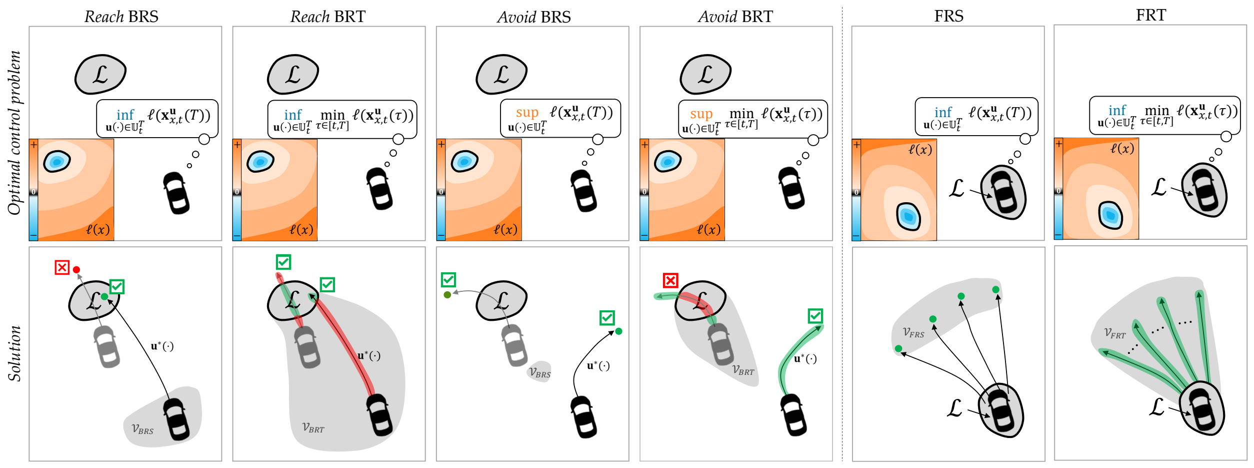

Reach BRS
Reach BRT
Avoid BRS
Avoid BRT
FRS
FRT
Optimal control problem
Solution
$\inf_{\mathbf{u}(\cdot)\in\mathbb{U}_t^T} \ell(\mathbf{x}_{x,t}^{\mathbf{u}}(T))$
$\inf_{\mathbf{u}(\cdot)\in\mathbb{U}_t^T} \min_{\tau\in[t,T]} \ell(\mathbf{x}_{x,t}^{\mathbf{u}}(\tau))$
$\sup_{\mathbf{u}(\cdot)\in\mathbb{U}_t^T} \ell(\mathbf{x}_{x,t}^{\mathbf{u}}(T))$
$\sup_{\mathbf{u}(\cdot)\in\mathbb{U}_t^T} \min_{\tau\in[t,T]} \ell(\mathbf{x}_{x,t}^{\mathbf{u}}(\tau))$
$\mathcal{L}$
$\ell(x)$
$\mathbf{u}^*(\cdot)$
$\mathcal{V}_{BRS}$
$\mathcal{V}_{BRT}$
$\mathcal{V}_{FRS}$
$\mathcal{V}_{FRT}$

Figure 2.2: (Top row) Optimal control problem for each variant of the reach and avoid sets. Inset image shows the level curves of the function $\ell(x)$ which encodes $\mathcal{L}$. (Bottom row) Grey region is the reachable set computed for initial time t. Optimal control trajectories are shown for two candidate initial conditions. Depending on if we were computing a set or a tube, we evaluate if the trajectory meets the constraint criterion at just the end of the trajectory, or over the entire length.

states x and all times, $\tau \in [t, T]$. While this will surely keep the system away from the undesirable states, the robot's only behavior will be avoidance. Can we use a different control law that is not avoidance-based (e.g., goal-seeking, minimum-energy, etc.) and use the safety-preserving control law only only when necessary?

Let $\Pi(\cdot) : \mathcal{O} \to \mathcal{U}$ be any mapping from states (where $\mathcal{O}$ is a state space, possibly different from $\mathbb{R}^n$) to controls. For example, the autonomous vehicle shown in Fig. 2.3 uses a deep neural network $\Pi(o)$ which takes in a monocular camera image $o \in \mathcal{O}$ and generates a speed and turn rate for the vehicle to navigate towards a desired room. Often times, Π will do an effective job of moving the robot towards the goal while avoiding collision with nearby obstacles, like the tables and chairs. However, Π has not been verified and is error-prone. For example, sometimes it decides to turn too aggressively, risking crashing the vehicle with a nearby chair leg. How do we leverage the benefits of the neural-network controller Π while ensuring it never collides?

Luckily, reachability can help us mitigate this problem. If we know where the obstacles are in the scene, then we can encodes them in our set of undesireable states, $\mathcal{L}$, and compute the corresponding avoid BRT, $\mathcal{V}_{BRT}$, and optimal safety-preserving controller $u^*(x, t)$.

Given all these components, our safety controller can be used in a *least-restrictive* fashion:

$$u(t) = \begin{cases} \Pi(o), & \text{if } x \in \mathcal{V}_{BRT}^{\mathsf{c}} \\ u^*(x,t), & \text{otherwise} \end{cases} \tag{2.53}$$

where $u^*(x, t)$ is the optimal safe controller as defined (2.52).

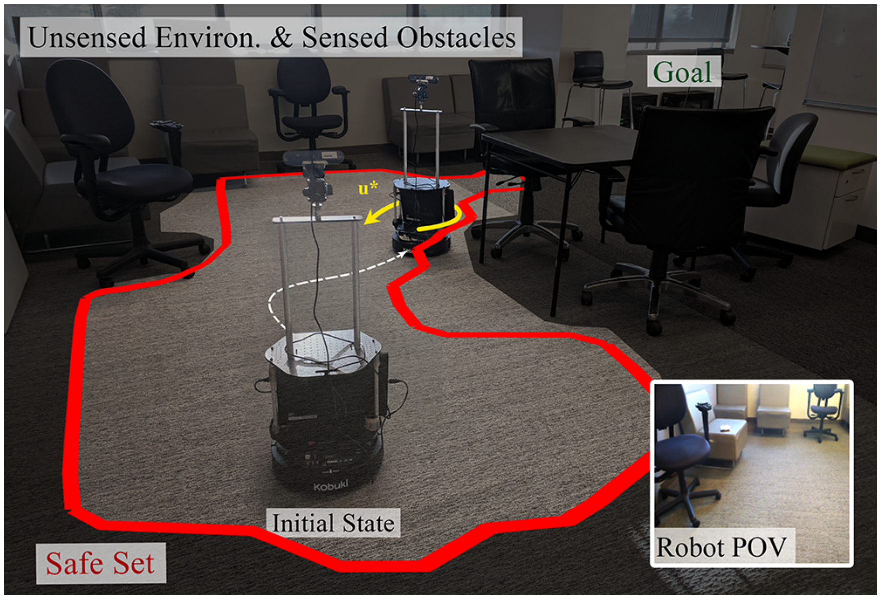


Figure 2.3: Visualization of how the safety-preserving optimal control (shown in yellow) from a BRT can be used to ensure safe robot navigation (steering robot away from collision with the chair) only at the boundary of the safe set (shown in red).

Where does $\mathcal{L}$ come from and what if we have to recompute our avoid set? So far we have assumed that we have access to $\mathcal{L}$ a priori, before the system is deployed. In some settings, this may make sense. For example, in the goal-reaching setting where an autonomous vehicle wants to park in the garage, it could be reasonable to assume that the end-user tells the autonomous car company where they live, and where their garage is, what dimensions it has, etc. Then, developing a controller to steer the car into the garage a priori is more straightforward for the AV engineer. However, in other cases it is difficult to know the precise shape of $\mathcal{L}$ before the robot is deployed. For example, the AV designer may not know a priori where the trash cans may be on a given day. Sometimes the trash cans are kept inside the garage, other times they are left on the street, other times they are blown into the driveway. If the AV engineer wants to encode the states occupied by the trashcans and develop a safety-preserving controller before the robot is deployed, this is incredibly challenging. Thus, it is necessary to (1) sense the undesireable states at runtime, (2) update the representation of $\mathcal{L}$, and (3) (re-)compute the safety-preserving control given the new information.

Handling a priori unknown, undesirable states $\mathcal{L}$ is a challenge beyond the scope of this dissertation. However, it has been studied in the broader autonomy community. A variety of mechanisms have been proposed to provide safety guarantees for systems using limited-range sensors to construct $\mathcal{L}$ incrementally [124, 130, 189, 114]. Although interesting results have emerged from these studies, the safety guarantees are provided by imposing specific assumptions on the sensor and/or the planner that are rather restrictive for a variety of real-world autonomous systems and sensors used for navigational purposes. In contrast, rather than proposing a specific planning and sensing paradigm that guarantees safety, we would like to design a safety framework that is compatible with a broad class of sensors, planners, and dynamics.

There are two main challenges with providing such a framework. The first challenge relates to ensuring safety with respect to unknown parts of the environment and external disturbances while minimally interfering with goal-driven behavior. Second, real-time safety assurances need to be provided as new environment information is acquired, which requires approximations that are both computationally efficient and not overly conservative. Moreover, this safety analysis should be applicable to a wide variety of real-world sensors, planners, and vehicles.

In [16], we proposed a safety framework that can overcome these challenges for autonomous vehicles operating in *a priori* unknown static environments under the assumption that the sensors work perfectly within their ranges. Erroneous and noisy sensors can make safety analysis significantly more challenging and we defer this to future work. Our framework is based on Hamilton Jacobi (HJ) reachability analysis [155, 158] and provides not only the set of states from which the dynamical system can always remain collision-free, but also provides an optimal controller that guarantees the system will never violate the state constraints. In particular, we treat the unknown environment at any given time as an obstacle and use HJ reachability to compute the *backward reachable tube (BRT)*, i.e. the set of states from which the vehicle can enter the unknown and potentially unsafe part of the environment, despite the best control effort. The complement of the BRT therefore represents the safe set for the vehicle. With this computation, we also obtain the corresponding least restrictive safety controller, which does not interfere with the planner unless the safety of the vehicle is at risk. Use of HJ reachability analysis in our framework thus allows us to overcome the first challenge—our framework can be applied to general nonlinear vehicles, sensors, and planners.

In general, due to the computationally expensive nature of HJ computations, this approach has not been leveraged in settings where the environment is not known beforehand and rather is sensed at run-time. To overcome this challenge, we propose a novel, real-time algorithm to compute the BRT. Our algorithm only locally updates the BRT in light of new environment information, which significantly alleviates the computational burden of HJ reachability while still maintaining the safety guarantees at all times. For more details on this work, please see [16].

2.3 From Dynamic Games to Multi-Agent Safety

So far, we have formalized optimal control and safety for robots operating in static environments. However, in reality, many robots will plan in the presence of other agents. For example, autonomous cars will navigate around other autonomous vehicles and human-driven cars; home robots will operate around people; factory robots will collaborate with human workers. However, so far our optimal control paradigm only allows for a single agent to be involved in decision-making. What if there are multiple agents making decisions and interacting? How is do we mathematically model this kind of interactive decision-making?

Cost Function	**Single-agent**	**Multi-Agent**
Static	Optimization	Game Theory
Time-evolution	Optimal Control	Differential *(cont.-time)* / Dynamic *(disc.-time)* Game

Table 2.2: Difference between optimization, optimal control, and game theory.

Dynamic game theory provides a framework for modelling and analyzing interaction in multi-agent dynamical systems. Intuitively, it can be thought of as the multi-agent extension to single-agent optimal control we have discussed so far. Dynamic game theory provides a set of mathematical tools for quantifying "optimal" decisions for each agent under various behavioral and informational assumptions. Note that dynamic games (also known as "difference games") typically refers to games subject to discrete-time dynamics equations. In this background section, we will begin with dynamic games only, but later introduce *differential games* (which are continuous-time games subject to differential equation dynamics) for the purpose of safety analysis. See Table 2.2 for a terminology comparison of games under various conditions.

In this section, we will also restrict our discussion to a particular type of game: *Stackelberg* games [218]. In these games, one agent takes the role of the *leader* and the other takes the role of the *follower*. The leader moves first, and the follower responds to the leader's decision. We will also summarize both the *open-loop* and *feedback* variants of Stackelberg games, which will be the most relevant for this dissertation. See Table 2.3 for a brief summary of relevant game-theory terminology. For an extensive treatment on information patterns in games, equilibrium concepts, and the continuous and discrete-time variants, please see [28].

Cost for agent A and B	$J_A \neq -J_B$	general-sum
	$J_A = -J_B$	zero-sum
State information	Access to only x^0	open-loop (OL)
	Access to any x^t	feedback
Order of play	Simultaneous	Nash equilibrium
	Leader-follower	Stackelberg equilibrium

Table 2.3: Key dynamic game terminology and the conditions under which the terms apply.

2.3.1 General-Sum Dynamic Games

Multi-Agent System Model. Since we have multiple agents, now our state x will represent the *joint* state of all agents. In this section we will consider two agents, A and B, with joint state:

$$x := [x_A, x_B]^\top. \tag{2.54}$$

Both agents have their own control inputs, $u_i \in \mathcal{U}_i$ $i \in \{A, B\}$, that affect the evolution of the joint dynamical system:

$$x^{t+1} = \tilde{f}(x^t, u_A^t, u_B^t). \tag{2.55}$$

Agent Cost Functions. Since we have multiple agents, we can model each of them as having their own decision-making objectives. For the purposes of this section, both agents are interested in minimizing their respective cost functions, J_i, which depend on the joint state and the actions of both agents. This is formally known as the *general-sum* game setting. However, as we will see later when establishing connections with safety analysis, sometimes it is beneficial to model the agents as sharing the same cost function where one agent is trying to minimize this cost and the other is trying to maximize.

Let's define the (discrete-time) cost function for each agent over time horizon $\{0, \dots, T\}$ as:

$$J_A(x^0, \mathbf{u}_A, \mathbf{u}_B) := \sum_{\tau=0}^{T-1} c_A(x^\tau, u_A^\tau, u_B^\tau) + \ell_A(x^T) \tag{2.56}$$

$$J_B(x^0, \mathbf{u}_A, \mathbf{u}_B) := \sum_{\tau=0}^{T-1} c_B(x^\tau, u_A^\tau, u_B^\tau) + \ell_B(x^T) \tag{2.57}$$

where the control sequences for each agent are $\mathbf{u}_A \in \mathbb{A}_0^{T-1}$ and $\mathbf{u}_B \in \mathbb{B}_0^{T-1}$.

Open-loop Stackelberg game. In open-loop dynamic games, both agents must plan their entire *control sequences*, given only the *initial condition*. In other words, this models an agent as determining their control sequence at the initial time, then "closing their eyes", and following it for the rest of the time horizon, without accounting for how the actual system state may evolve during control execution. More formally, a two-agent, general-sum open-loop (OL) Stackelberg game is:

$$\begin{aligned} \min_{\mathbf{u}_A \in \mathbb{A}_0^{T-1}} \quad & J_A(x^0, \mathbf{u}_A, \mathbf{u}_B^*(x^0, \mathbf{u}_A)) \\ \text{s.t.} \quad & x^{\tau+1} = \tilde{f}(x^\tau, u_A^\tau, u_B^\tau), \quad \forall \tau \in \{0, \dots, T-1\} \\ & \mathbf{u}_B^*(x^0, \mathbf{u}_A) = \arg\min_{\mathbf{u}_B \in \mathbb{B}_0^{T-1}} J_B(x^0, \mathbf{u}_A, \mathbf{u}_B) \\ & \qquad \text{s.t.} \quad x^{\tau+1} = \tilde{f}(x^\tau, u_A^\tau, u_B^\tau), \quad \forall \tau \in \{0, \dots, T-1\} \end{aligned} \tag{2.58}$$

Notice here how agent A optimizes over an entire control sequence $\mathbf{u}_A$ using only the current state and a best-response predictive model of how agent B will play their entire control sequence, $\mathbf{u}_B^*$.

We can explicitly define the value of this OL Stackelberg game starting from x^0 for agent A and agent B as:

$$V_A^0(x^0) = \min_{\mathbf{u}_A \in \mathbb{A}_0^{T-1}} J_A(x^0, \mathbf{u}_A, \mathbf{u}_B^*(x^0, \mathbf{u}_A)) \tag{2.59}$$

$$\text{where } \mathbf{u}_B^*(x^0, \mathbf{u}_A) = \arg\min_{\mathbf{u}_B} J_B(x^0, \mathbf{u}_A, \mathbf{u}_B)$$

$$V_B^0(x^0) = \min_{\mathbf{u}_B \in \mathbb{B}_0^{T-1}} J_B(x^0, \mathbf{u}_A^*(x^0), \mathbf{u}_B) \tag{2.60}$$

$$\text{where } \mathbf{u}_A^*(x^0) = \arg\min_{\mathbf{u}_A} J_A(x^0, \mathbf{u}_A, \mathbf{u}_B^*(x^0, \mathbf{u}_A))$$

Lets get a high-level understanding for one way to solve for these value functions. First, lets store the cumulative cost of agent B's objective for each possible agent A control sequence, $\mathbf{u}_A$. With this, we can search for the optimal control sequence for agent B *as a function of the initial condition and agent A's control sequence*: $\mathbf{u}_B^*(x^0, \mathbf{u}_A)$. Intuitively, this is a model how agent B can best respond to agent A's control sequence. Agent A can now use this model to plan their own optimal control sequence $\mathbf{u}_A^*(x^0)$, which is only a function of the initial condition, since agent A has a perfect model of how agent B will respond to all of their control sequences. We finally can obtain the optimal value of the game for each agent starting from x^0 by maximizing over their respective control sequences and plugging the models of how the other agent will behave.

Aside on OL Stackelberg computation. *It is straightforward to imagine computing the above for short time horizons and (small) discrete state and control spaces. For example, imagine* $t \in \{0, 1, 2, 3\}$ *and* $\mathcal{U}_i = \{\text{Up}, \text{Down}, \text{Left}, \text{Right}\}$. *Let the joint state be 2-dimensional, representing the stacked discrete x-position of agent A and agent B:* $x \in \mathcal{X} := \{[-5, -5], [-5, -4] \ldots, [5, 5]\}$ *where* $|\mathcal{X}| = 11 \times 11 = 121$. *Because the time horizon is short, we can feasibly enumerate all control sequences* $\mathbf{u}_A \in \mathbb{A}_0^2$ *and* $\mathbf{u}_B \in \mathbb{B}_0^2$ *(where the size of* $\mathbb{A}_0^2$ *and* $\mathbb{B}_0^2$ *is* $4^3 = 64$ *each). We can then store a large table for each* $(x^0, \mathbf{u}_A, \mathbf{u}_B)$ *tuple, where each entry contains the cost* J_A *or* J_B *of the overall joint system evolution. Here, the entire table storing* $J_i(x^0, \mathbf{u}_A, \mathbf{u}_B), i \in \{A, B\}$ *will be size* $121 \times 64 \times 64 = 495,616$. *Then we can choose the maximizing control sequences for each agent as defined above. However, as the time horizon T gets bigger and in the case that state and controls are continuous, we can no longer rely on a tabular, exhaustive way to solve this. Instead, we can turn to quasi-Newton methods such as L-BFGS [187, 9] or reformulate the bilevel optimization problem as a local, single-level optimization problem via the Karush-Kuhn-Tucker (KKT) stationarity conditions [194].*

Feedback Stackelberg game. In feedback dynamic games, both agents plan a *control policy*, which enables them to act optimally at *any future joint state*. In other words, this models the agents as able to accurately perceive the system state by always "having their eyes open", and constantly reacting to how the actual system state evolves during control execution.

To capture this reactivity in state and time, feedback Stackelberg games are naturally defined and solved recursively [93]. Let the values at the final time for a discrete-time, two-agent, general-sum feedback Stackelberg game be:

$$V_A^T(x^T) = \ell_A(x^T) \tag{2.61}$$
$$V_B^T(x^T) = \ell_B(x^T). \tag{2.62}$$

To make our notation simpler, let's define a new function called the Q-value, also called the state-action value function. Intuitively, this function will capture the immediate cost of taking a control action from the current state *plus the best possible future cost, assuming optimal play*. For example, the Q-value function for agent B is a map $Q_i : \mathbb{R}^n \times \mathcal{U}_A \times \mathcal{U}_B \to \mathbb{R}$ which captures the immediate cost of agent A choosing u_A and agent B choosing u_B from state x plus the best cost for the rest of the time horizon. Mathematically, we can write out each agent's Q-value function in the feedback Stackelberg game at any time $t \in \{0, \dots, T-1\}$ recursively:

$$Q_B^t(x^t, u_A^t, u_B^t) := c_B(x^t, u_A^t, u_B^t) + V_B^{t+1}(x^{t+1}) \tag{2.63}$$
$$Q_A^t(x^t, u_A^t) := c_A(x^t, u_A^t, \boldsymbol{\pi}_B^t(x^t, u_A^t)) + V_A^{t+1}(x^{t+1}) \tag{2.64}$$
$$\text{where } \boldsymbol{\pi}_B^t(x^t, u_A^t) = \arg\min_{u_B^t} \; Q_B^t(x^t, u_A^t, u_B^t).$$

Note that the Q-value function for agent A does not depend on agent B's control. This is because agent A is the *leader* in the Stackelberg game, and is assumed to have access to the best-response of agent B, π_B^t for each control that agent A may choose, u_A^t.

Finally, the value at any time $t \in \{0, \dots, T-1\}$ for agent A and agent B can be obtained via optimizing the Q-function at that time:

$$V_A^t(x^t) = \min_{u_A^t \in \mathcal{U}_A} \; Q_A^t(x^t, u_A^t) \tag{2.65}$$
$$V_B^t(x^t) = \min_{u_B^t \in \mathcal{U}_B} \; Q_B^t(x^t, \boldsymbol{\pi}_A^t(x^t), u_B^t) \tag{2.66}$$
$$\text{where } \boldsymbol{\pi}_A^t(x^t) = \arg\min_{u_A^t} \; Q_A^t(x^t, u_A^t)$$

Aside on feedback Stackelberg computation. *It should be apparent that the solution to the feedback Stackelberg game is a dynamic programming solution. This presents a significant computational challenge, since it suffers from the curse of dimensionality. Specifically, for discrete state, action, and time, its computational complexity is* $O(|\mathcal{X}| \times |\mathcal{U}_A| \times |\mathcal{U}_B| \times T)$.

2.3.2 Zero-Sum Differential Games

So far we have discussed a setting where each agent has their own objective that they are trying to minimize. However, there are other models we could use instead. For example, the agents could share the same objective $J_A = J_B$. Both agents could be minimizing this objective (i.e. cooperative dynamic game) or one agent could be minimizing while the other maximizing (i.e. adversarial dynamic game or *zero-sum* game). It turns out that the adversarial game models are crucial to traditional multi-agent safety methods! This is because they allow us to obtain *robust* controls for our robots, that enable our systems to achieve desirable properties despite a worst-case adversary.

Let's first mathematically define these zero-sum games. Let the joint state consisting of agent A and agent B's state be x as defined in (2.54). The continuous-time[2] joint dynamics are:

$$\dot{x} = f(x, u_A, u_B) \tag{2.67}$$

In zero-sum settings, both agents are concerned with the same cost function:

$$J(x, \mathbf{u}_A(\cdot), \mathbf{u}_B(\cdot), t) := \int_t^T c(x(\tau), u_A(\tau), u_B(\tau))d\tau + \ell(x(T)) \tag{2.68}$$

However, one agent is interested in minimizing while the other agent is interested in maximizing the cost. Formally, we can write this as:

$$\begin{aligned} V(x,t) := \inf_{\mathbf{u}_A(\cdot)\in\mathbb{A}_t^T} \sup_{\mathbf{u}_B(\cdot)\in\mathbb{B}_t^T} \quad & J(x, \mathbf{u}_A(\cdot), \mathbf{u}_B(\cdot), t) \\ \text{s.t.} \quad & \dot{x}(\tau) = f(x(\tau), u_A(\tau), u_B(\tau)), \quad \forall \tau \in [t, T] \\ & u_A(\tau) \in \mathcal{U}_A, u_B(\tau) \in \mathcal{U}_B \end{aligned} \tag{2.69}$$

where agent A seeks to minimize and agent B seeks to maximize the cost.

As posed right now, (2.69) describes an *open-loop* zero-sum game since the agent A must declare an entire control signal only given access to the initial state. Furthermore, it is a Stackelberg game because agent A is the leader and optimizing an entire control signal, while agent B is the follower who responds optimally to agent A with their own control signal. While this formulation may seem reasonable at first, it is actually incredibly pessimistic from the perspective of agent A. Agent A is in an incredibly difficult position, since they cannot adapt their control once the system begins evolving. In contrast, agent B has all the key information about how agent A will behave, enabling it to make an optimally adversarial decision during the optimization.

Could we come up with a less pessimistic formulation? What about a formulation that allows agent A to adapt their decisions as the system state evolves? In other words, can

[2]We switch to continuous-time notation here to make this subsection consistent with the formulation used in the zero-sum differential games used in HJ reachability, which is traditionally defined in continuous-time. However, discrete-time equivalents for this formulation are straightforward to write out.

we come up with a *feedback* variant of this zero-sum Stackelberg game? To do this, we actually need to modify how much information agent B has about agent A at any given time. Specifically, we will assume that agent B uses only *non-anticipative strategies* [216, 68]. Formally, the set of non-anticipative strategies $\mathfrak{B}_t^T$ for agent B is a collection of maps $\mathfrak{b} : \mathbb{A}_t^T \to \mathbb{B}_t^T$ such that agent B's strategy $\mathfrak{b}[\mathbf{u}_A]$ depends on $\mathbf{u}_A$. That is:

$$\begin{aligned} \mathfrak{b} \in \mathfrak{B}_t^T := \{&\mathfrak{b} : \mathbb{A}_t^T \to \mathbb{B}_t^T \ : \ \forall s \in [t,T], \forall \mathbf{u}_A(\cdot), \hat{\mathbf{u}}_A(\cdot) \in \mathbb{A}_t^T, \\ &(\mathbf{u}_A(\tau) = \hat{\mathbf{u}}_A(\tau) \ \text{ a.e. } \ \tau \in [t,s]) \Rightarrow (\mathfrak{b}[\mathbf{u}_A](\tau) = \mathfrak{b}[\mathbf{u}_A](\tau) \ \text{ a.e. } \ \tau \in [t,s])\}. \end{aligned} \tag{2.70}$$

The intuition for this definition is that agent B 's strategy ($\mathfrak{b}[\mathbf{u}_A]$) cannot start adapting to a change in agent A's control ($\mathbf{u}_A$) until such a change begins. Take a closer look at (2.70) to observe how this is captured mathematically. If two controls from agent A are identical, then agent B has to respond identically. In other words, agent B cannot respond differently to two agent A controls until they actually become different. Nevertheless, agent B still has *instantaneous informational advantage*, since at any time t it can still "see" agent A's choice of control and adapt its own accordingly (since $\mathfrak{b}[\mathbf{u}_A]$ is a function of $\mathbf{u}_A$).

Rewriting our differential game with this restriction on agent B's control strategy, we obtain a the following feedback, zero-sum differential game:

$$\begin{aligned} V(x,t) := \sup_{\mathfrak{b}\in\mathfrak{B}_t^T} \inf_{\mathbf{u}_A(\cdot)\in\mathbb{A}_t^T} \quad & J(x, \mathbf{u}_A(\cdot), \mathfrak{b}[\mathbf{u}_A](\cdot), t) \\ \text{s.t.} \quad & \dot{x}(\tau) = f(x(\tau), u_A(\tau), u_B(\tau)), \quad \forall \tau \in [t,T] \\ & u_A(\tau) \in \mathcal{U}_A, u_B(\tau) \in \mathcal{U}_B \end{aligned} \tag{2.71}$$

where agent B first chooses a strategy from the set of non-anticipative strategies and then agent A chooses its own control signal.

> ***Aside on discrete-time equivalent of* (2.71).** *At first, defining non-anticipative strategies can seem overly complicated. However, it is necessary to properly capture the notion of state-feedback when writing out the overall dynamic game where each player gets to choose continuous control signals. In discrete-time however, we can skirt around this definition by taking advantage of the discrete and sequential nature of decision-making. Specifically, we can write the discrete-time equivalent of* (2.71) *as:*
>
> $$\begin{aligned} V^t(x) := \min_{u_A^t} \max_{u_B^t} \ldots \min_{u_A^{T-1}} \max_{u_B^{T-1}} \quad & J^t(x, \mathbf{u}_A, \mathbf{u}_B) \\ \text{s.t.} \quad & x^{\tau+1} = \tilde{f}(x^\tau, u_A^\tau, u_B^\tau), \quad \forall \tau \in \{t, \ldots, T-1\} \end{aligned} \tag{2.72}$$
>
> *where the interleaved decisions of agent A and B capture both the feedback and "turn-taking" Stackelberg nature of the game.*

2.3.3 The Hamilton-Jacobi-Isaacs PDE & Multi-Agent Safety

Can we somehow solve for the optimal value function in (2.71)? It turns out that if we apply our knowledge of dynamic programming, we can follow a similar proof as earlier and show that the value function is the unique viscosity solution to the Hamilton-Jacobi-Isaacs (HJI) partial differential equation (PDE):

$$\begin{aligned} \frac{\partial V(x,t)}{\partial t} &+ \min_{u_A \in \mathcal{U}_A} \max_{u_B \in \mathcal{U}_B} \Big\{ c(x, u_A, u_B) + \nabla_x V(x,t) \cdot f(x, u_A, u_B) \Big\} = 0 \\ V(x,T) &= \ell(x). \end{aligned} \tag{2.73}$$

Given the value function, the optimal actions for each agent are:

$$u_A^*(x,t) = \arg \min_{u_A \in \mathcal{U}_A} \max_{u_B \in \mathcal{U}_B} \Big\{ c(x, u_A, u_B) + \nabla_x V(x,t) \cdot f(x, u_A, u_B) \Big\} \tag{2.74}$$

$$u_B^*(x,t) = \min_{u_A \in \mathcal{U}_A} \arg \max_{u_B \in \mathcal{U}_B} \Big\{ c(x, u_A, u_B) + \nabla_x V(x,t) \cdot f(x, u_A, u_B) \Big\} \tag{2.75}$$

> ***Aside on Bellman vs. Isaacs terminology.*** *Note that while before we referred to the equation as the Hamilton-Jacobi-*Bellman *equation, we refer to this variant as the Hamilton-Jacobi-*Isaacs *equation in honor of Rufus Isaacs. Isaacs pioneered pursuit-evasion games, proposing the principle of optimality for dynamic games in the 1950's at the same time as Richard Bellman proposed it for optimal control problems.*

Multi-Agent Safety via HJ Reachability

We now have a handle on how to solve zero-sum games between two agents. What if we wanted to guarantee that the joint system abides by some state constraints? For example, what if we want to come up with an optimal control which *guarantees* that our robot never collides with the other agent, despite the other agent's best efforts to crash? Or, an optimal control trajectory which guarantees that the robot reaches its goal despite the adversary attempting to prevent this? Just like when we studied safety analsyis for single-agent systems, we can use the tools from game theory and the HJI-PDE to formalize safety analysis of multi-agent systems. Specifically, let's write down the multi-agent variant of HJ reachability analysis.

Reach Problems (i.e., goal satisfaction)

We are going to start by considering *reach* problems again. Consider a shared autonomy system where the human controls a ground vehicle robot via a joystick and the robot can also input its own controls. Imagine that the robot wants to reach a goal location in a room, like its charging dock. A worst-case scenario would be that the human is trying to actively *prevent* the robot from reaching the goal. What set of initial states can we guaruntee that the robot can reach the goal despite the human's best efforts to steer the robot away from the goal? HJ reachability can answer this question for us yet again.

Mathematically, we will use the same notion of a target set $\mathcal{L} \subseteq \mathbb{R}^n$, however this is now defined in the *joint* state-space of the two agents. In our zero-sum setting, $\mathcal{L}$ defines the set of states that agent A wants to reach and agent B wants to prevent agent A from reaching.

The multi-agent variant of a *backwards reachable set (BRS)* and a *backwards reachable tube (BRT)* are defined:

$$\mathcal{V}_{BRS}(t) = \{x : \exists \mathbf{u}_A(\cdot) \in \mathbb{A}_t^T, \forall \mathbf{u}_B(\cdot) \in \mathbb{B}_t^T \mathbf{x}_{x,t}^{\mathbf{u}_A,\mathbf{u}_B}(T) \in \mathcal{L}\} \tag{2.76}$$

$$\mathcal{V}_{BRT}(t) = \{x : \exists \mathbf{u}_A(\cdot) \in \mathbb{A}_t^T, \forall \mathbf{u}_B(\cdot) \in \mathbb{B}_t^T, \exists \tau \in [t,T] \mathbf{x}_{x,t}^{\mathbf{u}_A,\mathbf{u}_B}(\tau) \in \mathcal{L}\}. \tag{2.77}$$

Intuitively, this is saying that if the system begins inside the BRS/BRT, $x \in \mathcal{V}$, then there exists a control signal that agent A can apply so it can get to the target set despite agent B's best-effort to prevent this.

Once again, we will define a Lipschitz function $\ell(x)$ such that our target set $\mathcal{L}$ is equal to the sub-zero level set of this function: $x \in \mathcal{L} \iff \ell(x) \leq 0$. The zero-sum game objective will be:

$$J(x, \mathbf{u}_A(\cdot), \mathbf{u}_B(\cdot), t) = \ell(\mathbf{x}_{x,t}^{\mathbf{u}_A,\mathbf{u}_B}(T)). \tag{2.78}$$

We can now modify our game formulation similarly to how we did in the single-agent optimal control setting and define the multi-agent BRS or BRT value functions:

BRS Value Function:

$$V(x,t) = \sup_{\mathfrak{b} \in \mathfrak{B}_t^T} \inf_{\mathbf{u}_A(\cdot) \in \mathbb{A}_t^T} \ell(\mathbf{x}_{x,t}^{\mathbf{u}_A,\mathbf{u}_B}(T)) \tag{2.79}$$

BRT Value Function:

$$V(x,t) = \sup_{\mathfrak{b} \in \mathfrak{B}_t^T} \inf_{\mathbf{u}_A(\cdot) \in \mathbb{A}_t^T} \min_{\tau \in [t,T]} \ell(\mathbf{x}_{x,t}^{\mathbf{u}_A,\mathbf{u}_B}(\tau)). \tag{2.80}$$

Then we can solve for these value functions via their corresponding HJI equation variants. Solving the following HJI-PDE allows us obtain the BRS value function:

$$\begin{aligned} &\frac{\partial V(x,t)}{\partial t} + \min_{u_A \in \mathcal{U}_A} \max_{u_B \in \mathcal{U}_B} \left\{ \nabla_x V(x,t) \cdot f(x, u_A, u_B) \right\} = 0 \\ &V(x,T) = \ell(x), \end{aligned} \tag{2.81}$$

and the HJI Variational Inequality's (VI) solution will be the BRT value function:

$$\begin{aligned} &\min \left\{ \ell(x) - V(x,t), \ \frac{\partial V(x,t)}{\partial t} + \min_{u_A \in \mathcal{U}_A} \max_{u_B \in \mathcal{U}_B} \left\{ \nabla_x V(x,t) \cdot f(x, u_A, u_B) \right\} \right\} = 0 \\ &V(x,T) = \ell(x). \end{aligned} \tag{2.82}$$

Note that the only difference between equation (2.81) and (2.24), and (2.82) and (2.41) is the inclusion of agent B's control input in the optimization.

Finally, the desired BRS or BRT can be extracted from the relevant value function by extracting the sub-zero level set:

$$\mathcal{V}(t) = \{x : V(x,t) \leq 0\}. \tag{2.83}$$

Avoid Problems (i.e., safety satisfaction)

Another incredibly important class of problems for safety analysis are avoid problems. Here we are concerned with the *backwards avoid set*, also known as the *avoid backwards reachable set (BRS)*, of a multi-agent dynamical system which is very similar to the single-agent variant. Recall that the avoid BRS is the set of states from which if joint system begins, it is doomed to enter into the undesirable set of states. Once again, our target set is $\mathcal{L} \subseteq \mathbb{R}^n$, however this is now capturing the set of *joint* states we seek to *avoid*. For example, assume that the state of each agent is their xy-position, $x_i \in \mathbb{R}^2, i \in \{A, B\}$. If we were solving a collision-avoidance problem, then we could define $\mathcal{L} := \{x : ||x_A - x_B||_2^2 \leq \epsilon\} \subseteq \mathbb{R}^4$. This means the undesirable states are where agent A and agent B are closer than a radius of $\sqrt{\epsilon}$. The avoid BRS and BRTs can be defined as:

$$\begin{aligned}\mathcal{V}_{BRS}(t) &= \{x : \forall \mathbf{u}_A(\cdot) \in \mathbb{A}_t^T, \exists \mathbf{u}_B(\cdot) \in \mathbb{B}_t^T,\ \mathbf{x}_{x,t}^{\mathbf{u}_A,\mathbf{u}_B}(T) \in \mathcal{L}\} \\ \mathcal{V}_{BRT}(t) &= \{x : \forall \mathbf{u}_A(\cdot) \in \mathbb{A}_t^T, \exists \mathbf{u}_B(\cdot) \in \mathbb{B}_t^T, \exists \tau \in [t,T],\ \mathbf{x}_{x,t}^{\mathbf{u}_A,\mathbf{u}_B}(\tau) \in \mathcal{L}\}.\end{aligned} \tag{2.84}$$

It turns out that capturing these sets via our zero-sum game is quite straightforward. Instead of modelling agent A as trying to minimize $\ell(\cdot)$ (and therefore try to enter $\mathcal{L}$) and agent B as maximizing ℓ, we will flip the agent's objectives. Agent A now wants to *maximize* $\ell(\cdot)$ (and therefore try to *stay outside of* the undesirable states $\mathcal{L}$), while agent B is trying to minimize the value and drive the system towards the undesirable states. This switch in roles is easy to see mathematically:

Avoid BRS Value Function:

$$V(x,t) = \inf_{\mathfrak{b} \in \mathfrak{B}_t^T} \sup_{\mathbf{u}_A(\cdot) \in \mathbb{A}_t^T} \ell(\mathbf{x}_{x,t}^{\mathbf{u}_A,\mathbf{u}_B}(T)) \tag{2.85}$$

Avoid BRT Value Function:

$$V(x,t) = \inf_{\mathfrak{b} \in \mathfrak{B}_t^T} \sup_{\mathbf{u}_A(\cdot) \in \mathbb{A}_t^T} \min_{\tau \in [t,T]} \ell(\mathbf{x}_{x,t}^{\mathbf{u}_A,\mathbf{u}_B}(\tau)). \tag{2.86}$$

To solve for these value functions, we can simply modify (2.81) and (2.82) by making agent A do $\max_{u_A}$ and agent B do $\min_{u_B}$. Like before, the desired multi-agent avoid BRS and BRT's can be extracted from the relevant value functions via the sub-zero level set.

Aside on zero-sum safety models for human-robot interaction. *This zero-sum assumption is good from a safety perspective because the guarantees and controllers we* *get out of this protect us against the worst-case. For example, if agent A is a plane's control system and agent B is a model of a wind disturbance, this worst-case model provides us strong safety guarantees against the highest possible winds that could shake our plane. However, when robots interact with people, this built-in adversarial assumption is often too pessimistic in practice. Humans aren't always optimal adversaries, nor seek to compete with the robot in a zero-sum fashion. Unfortunately, if robots are deployed with these worst-case safety controllers and guarantees, it leads to overly conservative robot behavior. However, if we relax this worst-case modelling assumption about human behavior, we risk compromising the quality of our safety analysis in the (unlikely but possible) event that the human is a true adversary. Addressing this fundamental safety challenge is core to this dissertation.*

2.4 Cost Function Design via Inverse Reinforcement Learning

When posing our optimal control problems or dynamic games, we have so far assumed that we know how to specify good cost functions, J, that capture all aspects of the behavior we care about. While it is true that sometimes we can design these effectively (e.g., find a control signal which minimizes fuel consumption), in other scenarios designing a "good" cost function is more challenging. For example, in our multi-agent scenario where agent A is the robot and agent B is a human, how do we know what agent B cares about? How should we accurately specify their cost, J_B? Or, consider designing an optimal controller for an autonomous car. There are so many aspects that a passenger in the vehicle may care about, from comfort to safety to efficiency. How can we possibly design J to capture all these unique aspects that the passenger will care about?

Let's think about how to tackle these questions. Before designing the cost function for the autonomous car that is driving a passenger around, consider letting the human drive the car around the neighborhood. By observing the human drive the car, we can glean insights onto how they like to drive: we can see that they always drive at low speeds, make smooth but efficient turns, and keep large safety margins between themselves and nearby obstacles or agents. The key idea here is

observations of an agent's behavior leak information about their objectives.

This idea of inferring the objectives from observations of agent behavior is formally called *Inverse Optimal Control (IOC)* [109] or *Inverse Reinforcement Learning (IRL)* [162]. IOC and IRL are two foundational inverse-problem frameworks from the control and machine learning communities, respectively. Conceptually, both approaches have the same goal: infer the objective of an agent by observing demonstrations (i.e., state and control trajectories) of the agent's behavior. However, there are slight technical differences between IOC and IRL: IOC assumes that the agent's behavior is generated by a stabilizing control law, while IRL assumes that the agent behaves optimally under their true objective. For a historical perspective on these variants of IOC/IRL, please see [1].

Since these foundational works, there have been many variants of the IRL problem formulation. In this dissertation, Maximum Entropy IRL [237] is a core method used throughout, so we will re-derive this variant of IRL as background. Note that we will use the AI notation of maximizing reward R in contrast to minimizing cost J like we have been doing so far.

2.4.1 Maximum Entropy IRL

Imagine that your friend demonstrates a behavior to you, like walking around an office room. Just by observing their trajectory (Fig. 2.4), can you infer how they chose that particular motion?

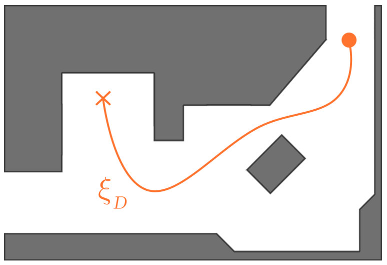


Figure 2.4: The demonstrator (orange dot) demonstrates a trajectory, ξ_D, by moving through a space. The observer seeks to discover what objective function the demonstrator is optimizing to produce this trajectory.

This is the typical Inverse Reinforcement Learning (IRL) problem, which seeks to explain an observed demonstration by uncovering the demonstrator's unknown objective function. Unfortunately, however, this is an ill-posed problem; many different objective functions can produce the same behavior and many different behaviors can be explained by the same objective function. Furthermore, demonstrations are often noisy.

Maximum Entropy IRL [237] addresses these issues by treating the demonstrations as observations drawn from some distribution that models the demonstrator as being *approximately optimal*. There are many candidate distributions, however, so the question remains: how do we choose the "best" one?

As we will derive here, the principle of maximum entropy allows us to find a distribution across trajectories that ensures we are not favoring any trajectories *other than the ones that are similar to the demonstrated one*. This also helps to resolve the ambiguity over objective functions inherent to the IRL formulation.

2.4.2 Matching Feature Counts

First, let's formalize how we measure relevant properties of the demonstrator's behavior. Let a trajectory $\xi \in \Xi$ with horizon T be a sequence of states and actions: $\xi = \{(x^1, u^1), (x^2, u^2), \ldots (x^T, u^T)\}$. Define a function, $f : \Xi \to \mathbb{R}^k$, that maps trajectories to a vector of real values. These real values, called *features*, can represent quantities that the agent might care about when producing its behavior; for example the agent's distance to obstacles, the speed of their movement, distance to other agents, etc.

Following prior work in IRL [3], we want to find distribution over observations (trajectories in this case), $P(\xi)$, that matches the empirical feature values in expectation

$$\mathbb{E}_{\xi \sim P(\xi)}[f(\xi)] = f_D$$

where $\xi \in \Xi$ is a trajectory and f_D are the feature values of the demonstration. These can be empirically computed from a set of demonstrated trajectories $D = \{\xi_1, \xi_2, \ldots, \xi_m\}$

$$f_D = \frac{1}{|D|} \sum_{\xi \in D} f(\xi)$$

Aside on matching the demonstration's feature counts. *Why do we want the feature expectations to match those of the demonstrated trajectories? Abbeel and Ng [3] shed light on why this is a good constraint to have. Let $r(x^t, u^t) = \lambda^\top f(x^t, u^t)$ be the reward of being in state x^t and executing u^t at time t and λ be a weight on the features of the state and action. Now, the expected return of executing a trajectory $\xi = \{(x_1, u_1), (x_2, u_2), \ldots (x_T, u_T)\}$ sampled from our distribution $P(\xi)$ is:*

$$\begin{aligned}
\mathbb{E}_{\xi \sim P(\xi)}\left[\sum_{t=0}^{T} r(x^t, u^t)\right] &= \mathbb{E}_{\xi \sim P(\xi)}\left[\sum_{t=0}^{T} \lambda^\top f(x^t, u^t)\right] \\
&= \lambda^\top \mathbb{E}_{\xi \sim P(\xi)}\left[\sum_{t=0}^{T} f(x^t, u^t)\right] \\
&= \lambda^\top \mathbb{E}_{\xi \sim P(\xi)}\left[f(\xi)\right]
\end{aligned}$$

where $f(\xi)$ is the sum of features of the state-action pairs encountered along the trajectory. We now see that the expected return of executing a trajectory can be written as a weighted feature expectation, $\lambda^\top \mathbb{E}_{\xi \sim P(\xi)}[f(\xi)]$. This means that if we have distributions that induce matching feature expectations, then they also produce trajectories that have matching returns in expectation!

2.4.3 Resolving Ambiguity via the Principle of Maximum Entropy

We want to make as few possible assumptions about the demonstrated trajectory while still matching the feature expectations. Recall that high entropy means high uncertainty. Thus, the key idea of Maximum Entropy IRL is that *finding the distribution that maximizes entropy over trajectories (subject to matching demonstrated feature values in expectation), we avoid favoring any particular trajectory other than ones that satisfy the feature constraints [237].*

Note that the idea of matching feature counts applies beyond just the Maximum Entropy IRL variant. However, what MaxEnt does is it takes a probabilistic viewpoint on the IRL problem and uses the principle of maximum entropy to resolve ambiguity in choosing a distribution over decisions.

2.4.4 Maximum Entropy IRL Derivation

Let's formulate this key idea as an optimization problem. As we said before, we want to find the distribution that maximizes the entropy subject to the feature expectations constraint, and the constraint that the distribution is a valid probability distribution

$$\begin{aligned} \underset{P}{\text{maximize}} \quad & \int -P(\xi)\log P(\xi) d\xi \\ \text{subject to} \quad & \mathbb{E}_{\xi\sim P(\xi)}[f(\xi)] = \int P(\xi) f(\xi) d\xi = f_D, \\ & \int P(\xi) d\xi = 1, \\ & P(\xi) \geq 0, \forall \xi \in \Xi \end{aligned} \tag{2.87}$$

For simplicity, we will first ignore the inequality constraint in Equation 2.87. We will later show that the solution trivially satisfies this constraint. Therefore, we form the Lagrangian, using multipliers λ and ν as:

$$\begin{aligned} \mathcal{L}(P,\lambda,\nu) &= \int -P(\xi)\log P(\xi)d\xi + \lambda^\top \left(\int P(\xi)f(\xi)d\xi - f_D \right) + \nu \left(\int P(\xi) d\xi - 1 \right) \\ &= \int -\left(P(\xi)\log P(\xi) + \lambda^\top P(\xi) f(\xi) + \nu P(\xi)\right) d\xi - \lambda^\top f_D - \nu. \end{aligned}$$

Notice how the Lagrangian is a functional (i.e. a function of a function). Therefore to solve for the maximum of this functional, we will employ calculus of variations. Specifically, by applying the Euler-Lagrange equation

$$\frac{\partial F}{\partial P}(\xi, P(\xi), P'(\xi)) - \frac{d}{d\xi}\frac{\partial F}{\partial P'}(\xi, P(\xi), P'(\xi)) = 0, \tag{2.88}$$

we can find the function $P^\star : \Xi \to [0,1]$ that optimizes our functional F, which in general may be a function of the vector ξ, the function P, and the derivative of that function P' with respect to ξ. Looking back at our Lagrangian, let's define:

$$F(\xi, P(\xi), P'(\xi)) = -P(\xi)\log P(\xi) + \lambda^\top P(\xi) f(\xi) + \nu P(\xi)$$

We can see that our Lagrangian $\mathcal{L}$ is a function of ξ and distribution $P(\xi)$ but does not depend on the first derivative $P'(\xi)$:

$$\mathcal{L}(P, \lambda, \nu) = \int F(\xi, P(\xi), P'(\xi))d\xi - \lambda^\top f_D - \nu$$

Since the $\lambda^\top f_D$ and ν are constants, then their subtraction simply shifts the function, but the optimum of the Lagrangian will remain unchanged. So without loss of generality we can let $\lambda^\top f_D = 0, \nu = 0$.

Finally, the full optimization problem we would like to solve is

$$(P^\star, \lambda^\star, \nu^\star) = \arg\min_{\lambda,\nu} \arg\max_{P} \mathcal{L}(P, \lambda, \nu).$$

Solving for $P^\star$

Using Equation 2.88, we can take the partial derivative with respect to P, set it equal to zero, and solve for $P^\star$.

$$\frac{\partial F}{\partial P}(\xi, P(\xi), P'(\xi)) = 0$$

First we take the partial derivative $\frac{\partial F}{\partial P}(\xi, P(\xi), P'(\xi))$ and then rearrange terms:

$$-\log P(\xi) - 1 + \lambda^\top f(\xi) + \nu = 0 \implies \log P(\xi) = \lambda^\top f(\xi) + \nu - 1$$

Finally, we solve for $P^\star$:

$$P^\star(\xi) \;=\; e^{\lambda^\top f(\xi)+\nu-1}$$

Now we can substitute our solution back into the Lagrangian to get

$$\begin{aligned}
\mathcal{L}(P^*, \lambda, \nu) &= \int -\left(e^{\lambda^\top f(\xi)+\nu-1}\right)\log\left(e^{\lambda^\top f(\xi)+\nu-1}\right) + \\
&\qquad \lambda^\top\left(e^{\lambda^\top f(\xi)+\nu-1}\right)f(\xi) + \nu\left(e^{\lambda^\top f(\xi)+\nu-1}\right)d\xi - \lambda^\top f_D - \nu \\
&= \int -\lambda^\top f(\xi)\left(e^{\lambda^\top f(\xi)+\nu-1}\right) + e^{\lambda^\top f(\xi)+\nu-1} - \nu e^{\lambda^\top f(\xi)+\nu-1} \\
&\qquad + \lambda^\top f(\xi)e^{\lambda^\top f(\xi)+\nu-1} + \nu e^{\lambda^\top f(\xi)+\nu-1}d\xi - \lambda^\top f_D - \nu \\
&= \int e^{\lambda^\top f(\xi)+\nu-1}d\xi - \lambda^\top f_D - \nu
\end{aligned}$$

Solving for $\nu^\star$

After finding $P^\star$, the dual function is $g(\lambda, \nu) = \mathcal{L}(P^\star, \lambda, \nu)$ with dual problem $\min_{\lambda,\nu} g(\lambda, \nu)$. To find $\nu^\star$ we follow a similar procedure

$$\frac{\partial \mathcal{L}}{\partial \nu} = 0 \implies e^{\nu-1}\int e^{\lambda^\top f(\xi)}d\xi - 1 = 0 \implies e^{-\nu} = \int e^{\lambda^\top f(\xi)-1}d\xi$$

Solving for ν, we get

$$\nu^\star = -\log\left(\int e^{\lambda^\top f(\xi)-1}\right)$$

Since we are interested in the probability distribution that maximizes this optimization problem, we can plug $\nu^\star$ into $P^\star(\xi)$

$$\begin{aligned}P^\star(\xi) &= e^{\lambda^\top f(\xi)-\log(\int e^{\lambda^\top f(\xi)-1}d\xi)-1} \\ &= \frac{e^{\lambda^\top f(\xi)-1}}{\int e^{\lambda^\top f(\tilde{\xi})-1}d\tilde{\xi}}\end{aligned}$$

We can now view our resulting probability distribution as parameterized by λ, where

$$P^\star(\xi;\lambda) = \frac{e^{\lambda^\top f(\xi)}}{\int e^{\lambda^\top f(\tilde{\xi})}d\tilde{\xi}}$$

Note that this solution satisfies the inequality constraint we had in Equation 2.87.

Solving for $\lambda^\star$

Above, we kept λ as a parameter under our distribution. The role of λ is to *weight* the various feature values of ξ. Given an observation of the demonstrator's trajectory, ξ_D, we want to choose λ so that it maximizes the likelihood of the observed trajectory in our distribution. To find an estimate of the λ that maximizes this likelihood, we can solve

$$\lambda^\star = \arg\max_\lambda \log P(\xi_D;\lambda) = \arg\max_\lambda \lambda^\top f(\xi_D) - \log\left(\int e^{\lambda^\top f(\xi)}d\xi\right)$$

Let $M = \lambda^\top f(\xi_D) - \log\left(\int e^{\lambda^\top f(\tilde{\xi})}d\tilde{\xi}\right)$. To find optimal λ, take the gradient of the objective w.r.t. λ

$$\begin{aligned}\nabla_\lambda M &= f(\xi_D) - \frac{1}{\int e^{\lambda^\top f(\tilde{\xi})}d\tilde{\xi}}\int f(\xi)e^{\lambda^\top f(\xi)}d\xi \\ &= f(\xi_D) - \int f(\xi)\frac{e^{\lambda^\top f(\xi)}d\xi}{\int e^{\lambda^\top f(\tilde{\xi})}d\tilde{\xi}} \\ &= f(\xi_D) - \int f(\xi)P(\xi;\lambda)d\xi \\ &= f(\xi_D) - \mathbb{E}_{\xi\sim P(\xi;\lambda)}[f(\xi)]\end{aligned}$$

This now gives us an intuitive gradient update rule, where we want to minimize the difference between the demonstrated feature values and the expected feature values under the estimated distribution. We can solve for λ by gradient descent, with the update rule

$$\lambda_{i+1} = \lambda_i + \alpha\left(f(\xi_D) - \mathbb{E}_{\xi\sim P(\xi;\lambda)}[f(\xi)]\right)$$

2.5 Human Modelling & Behavior Prediction

So far, we have been talking fairly abstractly about two agents interacting. However, the core of this dissertation is interested in scenarios where one agent is a robot and the other is a human. How does our modelling and control framework change when a human is one of the agents?

Fortunately, dynamical systems models still apply: we can still describe the evolution of the human and robot state via a function f that admits control inputs. But, how do we know what control inputs the human will choose when interacting with the robot? In other words, can we predict how the human will interact with the robot? Human behavior and motion prediction is an ever-expanding subdomain of robotics. For an incredibly helpful and in-depth survey, see [183]. Inspired by [183], we will break down the approaches to human modelling and behavior prediction into four categories: robust, physics-based, pattern-based, and planning-based models (see Table 2.4). This dissertation primarily utilizes the robust and planning-based models, however many of the core issues about human model misspecification and its effect on safe robot decision-making applies more broadly to any of the predictive human models summarized here.

Robust	predict all or worst-case behavior (e.g., FRS, multi-agent BRS/BRT)
Physics	assume simple $\mathbf{u}(\cdot)$ and simulate with f (e.g., velocity-obstacles)
Pattern	learn predictor conditioned on history (e.g., GMMs, GPs, HMMs, NNs)
Planning	model human as optimizing an objective (e.g., MDPs, opt. ctrl / games)

Table 2.4: Categories of human models used for behavior prediction.

Robust. When safety is of the utmost concern, using robust predictors can be desirable. For example, predicting *any* dynamically-feasible state the human could reach via the FRS enables the robot to avoid all potential collisions. In practice this unfortunately leads to overly conservative robot behavior. Not only does it assume that the human could go *anywhere*, but this also does not inherently account for the robot's ability to take safety maneuvers in response to the human behavior. Addressing the former issue (of modelling the human as able to go anywhere) is an active area of study [67, 14]. The later issue is something that game-theory and the BRS can help us handle. The multi-agent BRS accounts for how the robot can take evasive maneuvers to avoid an adversarial, collision-seeking human. While this is less conservative because now robot can react, it still models the human as an adversary who is willing to exert full control authority to antagonize the robot, which is still too pessimistic of a model for many human-robot interaction settings. Reducing conservatism of the human predictions while maintaining robustness is an active area of research, and one of the main subjects of this thesis.

Physics-based. These approaches use a physics-based dynamics model of the human and assume access to a simple control scheme. Together, these two enable the forward simulation of state trajectories. For example, a simple control scheme could be to assume the agent will continue moving with a constant velocity or constant acceleration. One such simple but foundational method is the reciprocal velocity obstacle [215]. However, the main challenge here is that these models typically only apply to low-dimensional systems and potentially fail to capture longer-horizon human reasoning (e.g., goal-driven behavior) or cooperative / adversarial multi-agent behavior.

Pattern-based. Popularized by the machine learning and computer vision communities, pattern-based methods leverage the availability of large datasets of human behavior (e.g., human driving data). These approaches fit different function approximators (such as neural networks (NNs) [188] or Gaussian Mixture Models (GMMs) [127]) to predict future state and action trajectories of the human conditioned on past snippets of behavior. These are particularly popular in industry, with AV companies like Waymo releasing an ever increasing quantity of pattern-based predictor variants [44, 232, 212]. An open challenge with these methods is their poor sample efficiency and their lack of robustness to out-of-distribution inputs and tail events.

Planning-based. Planning-based methods model the human as a rational actor, minimizing their internal objective function via an optimization or planning procedure. Unfortunately, the assumption that humans are perfect optimizers is quite strict. Instead, an alternative model captures human decision-making as noisily-optimal. In other words, these planning-based models capture that people will *most often* make decisions that maximize their objective. These *rationality-based* models have roots in econometrics and mathematical psychology, but have now been widely used throughout robotics to model how people behave in various contexts. One such model used throughout this dissertation is the *Boltzmann-rational model*[3] of human behavior [21, 237]. If the human is deciding on an entire control trajectory, $\mathbf{u}_H$, then the Boltmann-rational model [237] assigns exponentially more probability to the human executing $\mathbf{u}_H$ if it is assigned high utility by the function E:

$$P(\mathbf{u}_H \mid x) = \frac{e^{E(x,\mathbf{u}_H)}}{\int_{\mathbb{U}_0^{T-1}} e^{E(x,\tilde{\mathbf{u}})} d\tilde{\mathbf{u}}}. \tag{2.89}$$

In general, E can take many forms, but perhaps the easiest form to understand is the cumulative reward along the human's trajectory: $E(x, \mathbf{u}_H) := \sum_{t=0}^{T-1} r(x^t, u_H^t) + \ell(x^T)$. Alternatively, if we seek to model the one-step, noisily-optimal human decision-making [173], we can use the state-action value Q to obtain:

$$P(u_H \mid x) = \frac{e^{Q(x,u_H)}}{\int_{\mathcal{U}_H} e^{Q(x,u)} du}. \tag{2.90}$$

Note that the models above do not capture how the human's controls u_H depend on the robot's controls. Thankfully, game-theoretic models are also studied as part of planning-based models, and extensions of the above equations which account for the human reaction

[3]This model can be thought of as a special case of Luce's choice rule [149].

to the robot actions are straightforward to formulate. These predictive models exhibit more stable behavior when faced with out-of-distribution data [205], but they are still highly sensitive to the right model design (e.g., model of the human's cost function, choice of information-pattern between the human and robot, etc.).

Part I

Safe Robot Navigation Despite Imperfect Human Models

Part I focuses on safe robot navigation around humans when the robot's predictive human model may be *misspecified*. Though there has been much recent work in building predictive human models, no model is ever perfect: a person can always move unexpectedly, in a way that is not predicted or not assigned sufficient probability. Even if robot is maintaining uncertainty over various aspects of the human's behavior, a misspecified hypothesis space will not allow a robot to correctly update its human model because *no aspect* of the model can explain the human's behavior. In such cases, the robot may plan trajectories that appear safe but, in fact, lead to collision. The core idea in Part I is: rather than trust a model's predictions blindly, the robot should use the model's current predictive accuracy to inform the degree of confidence in its future predictions. In this line of work, we propose *confidence-aware* human models for safe robot planning. We quantify the notion of confidence-awareness by enabling the robot to infer online if it's human model could ever evolve to well-explain the observed human data. Coupling this idea with motion planning algorithms, we enable robots like autonomous cars, quadcopters, and small ground robots to automatically adapt their behavior around people from conservative to efficient based on estimated model confidence.

Chapter 3

Confidence-aware Human Models for Robot Planning

This chapter is based on the papers "Confidence-aware motion prediction for real-time collision avoidance" [77] and "Probabilistically Safe Robot Planning with Confidence-Based Human Predictions" [74], written in collaboration with Jaime Fisac, David Fridovich-Keil, Sylvia Herbert, Steven Wang, Claire Tomlin, and Anca Dragan.

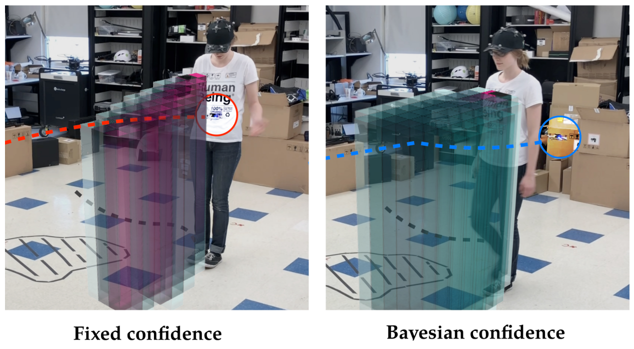


Figure 3.1: When planning around humans, accurate predictions of human motion (visualized here pink and blue, representing high and low probability respectively) are an essential prerequisite for safety. Unfortunately, these approaches may fail to explain all observed motion at runtime (e.g. human avoids unmodeled spill on the ground), leading to inaccurate predictions, and potentially, collisions (left). Our method addresses this by updating its *predictive model confidence* in real time (right), leading to more conservative motion planning in circumstances when predictions are known to be suspect.

Motion planning serves a key role in robotics, enabling robots to automatically compute trajectories that achieve the specified objectives while avoiding unwanted collisions. In many situations of practical interest, such as autonomous driving and UAV navigation, it is important that motion planning account not just for the current state of the environment, but also for its *predicted* future state. Often, certain objects in the environment may move in active, complex patterns that cannot be readily predicted using straightforward physics models; we shall refer to such complex moving objects as *agents*. Examples of agents considered in this work include pedestrians and human-driven vehicles. Predicting the future state of these agents is generally a difficult problem. Some of the key challenges include unclear and varying intents of other agents, mismatches between dynamics models and reality, incomplete sensor information, and interaction effects.

One popular approach to addressing the challenge of *a priori* unknown agent intent is to use rule-based or data-driven algorithms to predict *individual trajectories* for each agent, as in [190]. Alternatively, [238], [23], and [120] explicitly predict an agent's full state distribution over time; this representation may be better suited to capturing uncertainty in an agent's dynamics and the environment itself. [220] and [73] pose the prediction problem game-theoretically to model coupled human-robot interaction effects explicitly.

Unfortunately, a significant problem still remains: if an agent suddenly moves in a way that is not predicted, or not assigned sufficient probability, the robot may not react appropriately. For example, in Fig. 3.1 a pedestrian is walking around an obstacle which the robot, a quadcopter, cannot detect. To the robot, such behavior may be assigned very low probability, which could lead the robot to plan a dangerous trajectory. In this particular example, this inaccuracy caused the quadcopter to collide with the pedestrian (Fig. 3.1, left).

To prepare for this eventuality, we introduce the idea of *confidence-aware prediction*. We argue that, in addition to predicting the future state of an agent, it is also crucial for a robot to assess the quality of the mechanism by which it is generating those predictions. That is, a robot should reason about how *confident* it is in its predictions of other agents before attempting to plan future motion. For computational efficiency, the quadcopter uses a simplified model of pedestrian dynamics and decision making. Thus equipped, it generates a time-varying probability distribution over the future state of the pedestrian, and plans trajectories to a pre-specified goal that maintain a low probability of collision. Fig. 3.1 (right) illustrates how this approach works in practice. The quadcopter maintains a Bayesian *belief* over its prediction confidence. As soon as the pedestrian moves in a way that was assigned low probability by the predictive model, the quadcopter adjusts its belief about the accuracy of that model. Consequently, it is less certain about what the pedestrian will do in the future. This leads the quadcopter's onboard motion planner, which attempts to find efficient trajectories with low probability of collision, to generate more cautious—and perhaps less efficient—motion plans.

In order to improve the robustness of generated motion plans, we employ the recent FaSTrack framework from [95] for fast and safe motion planning and tracking. FaSTrack quantifies the maximum possible tracking error between a high-order dynamical model of the physical robot and the (potentially lower-order) dynamical model used by its motion planner. Solving an offline Hamilton-Jacobi reachability problem yields a guaranteed *tracking error bound* and the corresponding *safety controller*. These may be used by an out-of-the-box real-time motion planning algorithm to facilitate motion plans with strong runtime collision-avoidance guarantees.

The remainder of this work is organized as follows. Section 3.1 places this work in the context of existing literature in human motion modeling and prediction, as well as robust motion planning. Section 3.2 frames the prediction and planning problems more formally, and introduces a running example used throughout the work. Section 3.3 presents our main contribution: confidence-aware predictions. Section 3.4 showcases confidence-aware predictions in operation in several examples. Section 3.5 describes the application of the robust motion planning framework from FaSTrack to this setting, in which predictions are probabilistic. Section 3.6 explores a connection between our approach and reachability theory. Section 3.7.1 presents experimental results from a hardware demonstration. Finally, Section 3.8 concludes with a discussion of some of the limitations of our work and how they might be addressed in specific applications, as well as suggestions for future research.

3.1 Prior Work

3.1.1 Human Modeling and Prediction

One common approach for predicting human actions is to collect data from real-world scenarios and train a machine learning model via supervised learning. Such techniques use the human's current state, and potentially her prior state and action history, to predict future actions directly. [8], [62], [123], [133], and [94] demonstrate the effectiveness of this approach for inference and planning around human arm motion. Additionally, [94] focus on multi-step tasks like assembly, and [190] and [67] address the prediction problem for human drivers.

Rather than predicting actions directly, an alternative is for the robot to model the human as a rational agent seeking to maximize an unknown objective function. The human's actions up to a particular time may be viewed as Bayesian evidence from which the robot may infer the parameters of that objective. Assuming that the human seeks to maximize this objective in the future, the robot can predict her future movements, e.g. [162], [21], [238], and [13]. In this paper, we build on this work by introducing a principled online technique for estimating confidence in such a learned model of human motion.

3.1.2 Safe Robot Motion Planning

Once armed with a predictive model of the human motion, the robot may leverage motion planning methods that plan around uncertain moving obstacles and generate real-time dynamically feasible and safe trajectories.

To avoid moving obstacles in real time, robots typically employ reactive and/or path-based methods. Reactive methods directly map sensor readings into control, with no memory involved, e.g. [29]. Path-based methods such as rapidly-exploring random trees from [110] and A* from [92] find simple kinematic paths through space and, if necessary, time. These path-based methods of planning are advantageous in terms of efficiency, yet, while they have in some cases been combined with probabilistically moving obstacles as in [10, 238], they do not consider the endogenous dynamics of the robot or exogenous disturbances such as wind. As a result, the robot may deviate from the planned path and potentially collide with obstacles. It is common for these plans to try to avoid obstacles by a heuristic margin of error. [95, 78] propose FaSTrack, a recent algorithm that provides a guaranteed tracking error margin and corresponding error-feedback controller for dynamic systems tracking a generic planner in the presence of bounded external disturbance. Our work builds upon FaSTrack to create an algorithm that can safely and dynamically navigate around uncertain moving obstacles in real time.

3.2 Problem Setup

We consider a single mobile robot operating in a shared space with a single human agent (e.g. a pedestrian or human-driven car). For simplicity, we presume that the robot has full knowledge of its own state and that of the human, although both would require online estimation in practice. As we present each formal component of this problem, we will provide a concrete illustration using a running example in which a quadcopter is navigating around a pedestrian.

3.2.1 Dynamical System Models and Safety

We will model the motion of both the human and the robot as the evolution of two dynamical systems. Let the state of the human be $x_H \in \mathbb{R}^{n_H}$, where n_H is the dimension of the human state space. We similarly define the robot's state, for planning purposes, as $x_R \in \mathbb{R}^{n_R}$. In general, these states could represent the positions and velocities of a mobile robot and a human in a shared environment, the kinematic configurations of a human and a robotic manipulator in a common workspace, or the positions, orientations, and velocities of human-driven and autonomous vehicles in an intersection. We express the evolution of these states over time as a family of ordinary differential equations:

$$\dot{x}_H = f_H(x_H, u_H) \,, \qquad \dot{x}_R = f_R(x_R, u_R) \tag{3.1}$$

where $u_H \in \mathbb{R}^{m_H}$ and $u_R \in \mathbb{R}^{m_R}$ are the control actions of the human and robot, respectively.

Running example: *We introduce a running example for illustration purposes throughout the paper. In this example we consider a small quadcopter that needs to fly to goal location $g_R \in \mathbb{R}^3$ in a room where a pedestrian is walking. For the purposes of planning, the quadcopter's 3D state is given by its position in space $x_R = [p_x, p_y, p_z]$, with velocity controls assumed decoupled in each spatial direction, up to $v_R = 0.25$ m/s. The human can only move by walking and therefore her state is given by planar coordinates $x_H = [h_x, h_y]$ evolving as $\dot{x}_H = [v_H \cos u_H, v_H \sin u_H]$. Intuitively, we model the human as moving with a fixed speed and controlling her heading angle. At any given time, the human is assumed to either move at a leisurely walking speed ($v_H \approx 1$ m/s) or remain still ($v_H \approx 0$).*

Ultimately, the robot needs to plan and execute an efficient trajectory to a pre-specified goal state (g_R), without colliding with the human. We define the keep-out set $\mathcal{K} \subset \mathbb{R}^{n_H} \times \mathbb{R}^{n_R}$ as the set of joint robot-human states to be avoided (for example, because they imply physical collisions). To avoid reaching this set, the robot must reason about the human's future motion when constructing its own motion plan.

Running example: *In our quadcopter-avoiding-pedestrian example, $\mathcal{K}$ consists of joint robot-human states in which the quadcopter is flying within a square of side length $l = 0.3$ m centered around the human's location, while at any altitude, as well as any joint states in which the robot is outside the environment bounds defined as a box with a square base of side $L = 3.66$ m and height $H = 2$ m, regardless of the human's state.*

3.2.2 Robust Robot Control

Provided an objective and a dynamics model, the robot must generate a motion plan which avoids the keep-out set $\mathcal{K}$. Unfortunately, this safety requirement is difficult to meet during operation for two main reasons:

1. *Model mismatch.* The dynamical system model f_R will never be a perfect representation of the real robot. This mismatch could lead to unintended collision.

2. *Disturbances.* Even with a perfect dynamics model, there may be unobserved, external "disturbance" inputs such as wind or friction. Without accounting for these disturbances, the system is not guaranteed to avoid $\mathcal{K}$, even if the planned trajectory is pointwise collision-free.

To account for modelling error and external disturbances, we could in principle design a higher fidelity dynamical model directly in a robust motion planning framework. Unfortunately, however, real-time trajectory optimization in high dimensions can be computationally burdensome, particularly when we also require some notion of robustness to external disturbance. Ideally we would like to enjoy the computational benefits of planning with a lower-fidelity model while enforcing the safety constraints induced by the higher-fidelity model. To characterize this model mismatch, we consider a higher fidelity and typically higher-order dynamical representation of the robot, with state representation $s_R \in \mathbb{R}^{n_S}$. This dynamical model will also explicitly account for external disturbances as unknown bounded inputs, distinct from control inputs. In order to map between this higher fidelity "tracking" state s_R and the lower fidelity "planning" state x_R, we shall assume a known projection operator $\pi : \mathbb{R}^{n_S} \to \mathbb{R}^{n_R}$. Fortunately, we can plan in the lower-dimensional state space at runtime, and guarantee robust collision avoidance via an offline reachability analysis that quantifies the effects of model mismatch and external disturbance. This framework, called FaSTrack and first proposed by [95], is described in further detail in Section 3.5.

Running example: *We model our quadcopter with the following flight dynamics (in the near-hover regime, at zero yaw with respect to a global coordinate frame):*

$$\begin{bmatrix} \dot{p}_x \\ \dot{p}_y \\ \dot{p}_z \end{bmatrix} = \begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix} , \begin{bmatrix} \dot{v}_x \\ \dot{v}_y \\ \dot{v}_z \end{bmatrix} = \begin{bmatrix} a_g \tan u_\theta \\ -a_g \tan u_\phi \\ u_T - a_g \end{bmatrix} , \tag{3.2}$$

where $[p_x, p_y, p_z]$ is the quadcopter's position in space and $[v_x, v_y, v_z]$ is its velocity expressed in the fixed global frame. We model its control inputs as thrust acceleration u_T and attitude angles (roll u_ϕ and pitch u_θ), and denote the acceleration due to gravity as a_g. The quadcopter's motion planner generates nominal kinematic trajectories in the lower-dimensional $[p_x, p_y, p_z]$ position state space. Therefore we have a linear projection map $\pi(s_R) = [I_3, 0_3]s_R$, that is, x_R retains the position variables in s_R and discards the velocities.

3.2.3 Predictive Human Model

In order to predict the human's future motion, the robot uses its internal model of human dynamics, f_H. Under this modeling assumption, the human's future trajectory depends upon the choice of control input over time, $u_H(\cdot)$. Extensive work in econometrics and cognitive science, such as [217, 149, 21], has shown that human behavior—that is, u_H—can be well modeled by utility-driven optimization. Thus, the robot models the human as optimizing a reward function, $r_H(x_H, u_H; \theta)$, that depends on the human's state and action, as well as a set of parameters θ. This reward function could be a linear combination of features as in many inverse optimal control implementations (where the goal or feature weighting θ must be learned, either online or offline), or more generally learned through function approximators such as deep neural networks, where θ are the trained weights as in [70].

We assume that the robot has a suitable human reward function r_H, either learned offline from prior human demonstrations or otherwise encoded by the system designers. Thus endowed with r_H, the robot can model the human's choice of control action as a probability distribution over actions conditioned on state. Under maximum-entropy assumptions ([237]) inspired by noisy-rationality decision-making models ([21]), the robot models the human as more likely to choose (discrete) actions u_H with high expected utility, in this case the state-action value (or Q-value):

$$P(u_H \mid x_H; \beta, \theta) = \frac{e^{\beta Q_H(x_H, u_H; \theta)}}{\sum_{\tilde{u}} e^{\beta Q_H(x_H, \tilde{u}; \theta)}} . \tag{3.3}$$

We use a temporally- and spatially-discretized version of human dynamics, $\tilde{f}_H$. These discrete-time dynamics may be found by integrating f_H over a fixed time step of Δt with fixed control u_H over the interval. Section **??** provides further details on this discretization.

Running example: *The quadcopter's model of the human assumes the human intends to reach some target location $g_H \in \mathbb{R}^2$ in a straight line. The human's reward function is given by the distance traveled over time step Δt, i.e. $r_H(x_H, u_H; g_H) = -v_H \Delta t$, and human trajectories are constrained to terminate at g_H. The state-action value, parameterized by $\theta = g_H$,* *captures the optimal cost of reaching g_H from x_H when initially applying u_H for a duration Δt:* $Q_H(x_H, u_H; g_H) = -v_H \Delta t - \|x_H + v_H \Delta t [\cos u_H, \sin u_H]^\top - g_H\|_2$.

Often, the coefficient β is termed the *rationality coefficient*, since it quantifies the degree to which the robot expects the human's choice of control to align with its model of utility. For example, taking $\beta \downarrow 0$ yields a model of a human who appears "irrational," choosing actions uniformly at random and completely ignoring the modeled utility. At the other extreme, taking $\beta \uparrow \infty$ corresponds to a "perfectly rational" human, whose actions exactly optimize the modeled reward function. As we will see in Section **??**, β can also be viewed as a measure of the robot's *confidence* in the predictive accuracy of Q_H.

Note that $Q_H(x_H, u_H; \theta)$ only depends on the human state and action and not on the robot's. Thus far, we have intentionally neglected discussion of human-robot interaction effects. These effects are notoriously difficult to model, and the community has devoted a significant effort to building and validating a variety of models, e.g. [214], [187]. In that spirit, we could have chosen to model human actions u_H as dependent upon robot state x_R in (3.3), and likewise defined Q_H to depend upon x_R. This extended formulation is sufficiently general as to encompass all possible (Markov) interaction models. However, in this work we explicitly do not model these interactions; indeed, one of the most important virtues of our approach is its robustness to precisely these sorts of modeling errors.

3.2.4 Probabilistically Safe Motion Planning

Ideally, the robot's motion planner should generate trajectories that reach a desired goal state efficiently, while maintaining safety. More specifically, in this context "safety" indicates that the physical system will never enter the keep-out set $\mathcal{K}$ during operation, despite human motion and external disturbances. That is, we would like to guarantee that $(\pi(s_R), x_H) \notin \mathcal{K}$ for all time.

To make this type of strong, deterministic, *a priori* safety guarantee requires the robot to avoid the set of all human states x_H which could possibly be occupied at a particular time, i.e. the human's *forward reachable set*. If the robot can find trajectories that are safe for *any* possible human trajectory then there is no need to predict the human's next action. Unfortunately, the forward reachable set of the human often encompasses such a large volume of the workspace that it is impossible for the robot to find a guaranteed safe trajectory to the goal state. This motivates refining our notion of prediction: rather than reasoning about all the places where the human *could* be, the robot can instead reason about *how likely* the human is to be at each location. This probabilistic reasoning provides a guide for planning robot trajectories with a quantitative degree of safety assurance.

Our probabilistic model of human control input (3.3) coupled with dynamics model f_H allows us to compute a probability distribution over human states for every future time. By relaxing our conception of safety to consider only collisions which might occur with sufficient probability P_{th}, we dramatically reduce the effective volume of this set of future states to avoid. In practice, P_{th} should be chosen carefully by a system designer in order to trade off overall collision probability with conservativeness in motion planning.

The proposed approach in this paper follows two central steps to provide a quantifiable, high-confidence collision avoidance guarantee for the robot's motion around the human. In Section 3.3 we present our proposed Bayesian framework for reasoning about the uncertainty inherent in a model's prediction of human behavior. Based on this inference, we demonstrate how to generate a real-time probabilistic prediction of the human's motion over time. Next, in Section 3.5 we extend a state-of-the-art, provably safe, real-time robotic motion planner to incorporate our time-varying probabilistic human prediction.

3.3 Confidence-Aware Human Motion Prediction

Any approach to human motion prediction short of computing a full forward reachable set must, explicitly or implicitly, reflect a model of human decision-making. In this work, we make that model explicit by assuming that the human chooses control actions in a Markovian fashion according to the probability distribution (3.3). Other work in the literature, such as [190], aims to learn a generative probabilistic model for human trajectories; implicitly, this training procedure distills a model of human decision making. Whether explicit or implicit, these models are by nature imperfect and liable to make inaccurate predictions eventually. One benefit of using an explicit model of human decision making, such as (3.3), is that we may reason directly and succinctly about its performance online.

In particular, the entropy of the human control distribution in (3.3) is a decreasing function of the parameter β. High values of β place more probability mass on high-utility control actions u_H, whereas low values of β spread the probability mass more evenly between different control inputs, regardless of their modeled utility Q_H. Therefore, β naturally quantifies how well the human's motion is expected to agree with the notion of optimality encoded in Q_H. The commonly used term "rationality coefficient", however, seems to imply that discrepancies between the two indicate a failure on the human's part to make the "correct" decisions, as encoded by the modeled utility. Instead, we argue that these inevitable disagreements are primarily a result of the model's inability to fully capture the human's behavior. Thus, instead of conceiving of β as a rationality measure, we believe that β can be given a more pragmatic interpretation related to the accuracy with which the robot's model of the human is able to explain the human's motion. Consistently, in this paper, we refer to β as *model confidence*.

An important related observation following from this interpretation of β is that the predictive accuracy of a human model is likely to change over time. For example, the human may change their mind unexpectedly, or react suddenly to some aspect of the environment that the robot is unaware of. Therefore, we shall model β as an unobserved, time-varying parameter. Estimating it in real-time provides us with a direct, quantitative summary of the degree to which the utility model Q_H explains the human's current motion. To do this, we maintain a Bayesian *belief* about the possible values of β. Initially, we begin with a uniform prior over β and over time this distribution evolves given measurements of the human's state and actions.

3.3.1 Real-time Inference of Model Confidence

We reason about the model confidence β as a hidden state in a hidden Markov model (HMM) framework. The robot starts with a prior belief b_-^0 over the initial value of β. In this work, we use a uniform prior, although that is not strictly necessary. At each discrete time step $k \in \{0, 1, 2, \dots\}$, it will have some belief about model confidence $b_-^k(\beta)$. [1] After observing a human action u_H^k, the robot will update its belief to b_+^k by applying Bayes' rule.

The hidden state may evolve between subsequent time steps, accounting for the important fact that the predictive accuracy of the human model may change over time as unmodeled factors in the human's behavior become more or less relevant. Since by definition we do not have access to a model of these factors, we use a naive "ϵ-static" transition model: at each time k, β may, with some probability ϵ, be re-sampled from the initial distribution b_-^0, and otherwise retains its previous value. We define the belief over the next value of β (denoted by β') as an expectation of the conditional probability $P(\beta' \mid \beta)$, i.e. $b_-^k(\beta') := \mathbb{E}_{\beta \sim b_+^{k-1}}[P(\beta' \mid \beta)]$. Concretely, this expectation may be computed as

$$b_-^k(\beta') = (1 - \epsilon) b_+^{k-1}(\beta') + \epsilon b_-^0(\beta') \ . \tag{3.4}$$

By measuring the evolution of the human's state x_H over time, we assume that, at every time step k, the robot is able to observe the human's control input u_H^k. This observed control may be used as evidence to update the robot's belief b_-^k about β over time via a Bayesian update:

$$b_+^k(\beta) = \frac{P(u_H^k \mid x_H^k; \beta, \theta) b_-^k(\beta)}{\sum_{\tilde{\beta}} P(u_H^k \mid x_H^k; \tilde{\beta}, \theta) b_-^k(\tilde{\beta})} \ , \tag{3.5}$$

with $b_+^k(\beta) := P(\beta \mid x_H^{0:k}, u_H^{0:k})$ for $k \in \{0, 1, \dots\}$, and $P(u_H^k \mid x_H^k; \beta, \theta)$ given by (3.3).

It is critical to be able to perform this update rapidly to facilitate real-time operation; this would be difficult in the original continuous hypothesis space $\beta \in [0, \infty)$, or even in a large discrete set. Fortunately, our software examples in Section 3.4 and hardware demonstration in Section 3.7.1 suggest that maintaining a Bayesian belief over a relatively small set of $N_\beta = 5$ discrete values of β distributed on a log scale achieves significant improvement relative to using a fixed value.

The "ϵ-static" transition model leads to the desirable consequence that old observations of the human's actions have a smaller influence on the current model confidence distribution than recent observations. In fact, if no new observations are made, successively applying time updates asymptotically contracts the belief towards the initial distribution, that is, $b_-^k(\cdot) \to b_-^0(\cdot)$. The choice of parameter ϵ effectively controls the rate of this contraction, with higher ϵ leading to more rapid contraction.

[1] To avoid confusion between discrete and continuous time, we shall use superscripts to denote discrete time steps (e.g. x_H^k) and parentheticals to denote continuous time (e.g. $x_H(t)$).

3.3.2 Human motion prediction

Equipped with a belief over β at time step k, we are now able to propagate the human's state distribution forward to any future time via the well-known Kolmogorov forward equations, recursively. In particular, suppose that we know the probability that the human is in each state x_H^κ at some future time step κ. We know that (according to our utility model) the probability of the human choosing control u_H^κ in state x_H^κ is given by (3.3). Accounting for the otherwise deterministic dynamics model $\tilde{f}_H$, we obtain the following expression for the human's state distribution at the following time step $\kappa + 1$:

$$P(x_H^{\kappa+1}; \beta, \theta) = \sum_{x_H^\kappa, u_H^\kappa} P(x_H^{\kappa+1} \mid x_H^\kappa, u_H^\kappa; \beta, \theta) \cdot \\ P(u_H^\kappa \mid x_H^\kappa; \beta, \theta) P(x_H^\kappa; \beta, \theta) \ , \tag{3.6}$$

for a particular choice of β. Marginalizing over β according to our belief at the current step time k, we obtain the overall occupancy probability distribution at each future time step κ:

$$P(x_H^\kappa; \theta) = \mathbb{E}_{\beta \sim b^k} P(x_H^\kappa; \beta, \theta) \ . \tag{3.7}$$

Note that (3.6) is expressed more generally than is strictly required. Indeed, because the only randomness in dynamics model $\tilde{f}_H$ originates from the human's choice of control input u_H, we have $P(x_H^{\kappa+1} \mid x_H^\kappa, u_H^\kappa; \beta, \theta) = \mathbb{1}\{x_H^{\kappa+1} = \tilde{f}_H(x_H^\kappa, u_H^\kappa)\}$.

3.3.3 Model Confidence with Auxiliary Parameter Identification

Thus far, we have tacitly assumed that the only unknown parameter in the human utility model (3.3) is the model confidence, β. However, often one or more of the auxiliary parameters θ are also unknown. These auxiliary parameters could encode one or more human goal states or intents, or other characteristics of the human's utility, such as her preference for avoiding parts of the environment. Further, much like model confidence, they may change over time.

In principle, it is possible to maintain a Bayesian belief over β and θ jointly. The Bayesian update for the hidden state (β, θ) is then given by

$$b_+^k(\beta, \theta) = \frac{P(u_H^k \mid x_H^k; \beta, \theta) b_-^k(\beta, \theta)}{\sum_{\tilde{\beta}, \tilde{\theta}} P(u_H^k \mid x_H^k; \tilde{\beta}, \tilde{\theta}) b_-^k(\tilde{\beta}, \tilde{\theta})} \ , \tag{3.8}$$

with $b_+^k(\beta, \theta) := P(\beta, \theta \mid x_H^{0:k}, u_H^{0:k})$ the running posterior and $b_-^k(\beta, \theta) := P(\beta, \theta \mid x_H^{0:k-1}, u_H^{0:k-1})$ the prior at time step k.

This approach can be practical for parameters taking finitely many values from a small, discrete set, e.g. possible distinct modes for a human driver (distracted, cautious, aggressive). However, for certain scenarios or approaches it may not be practical to maintain a full Bayesian belief on the parameters θ. In such cases, it is reasonable to replace the belief over θ with a point estimate $\bar{\theta}$, such as the maximum likelihood estimator or the mean, and substitute that estimate into (3.6). Depending on the complexity of the resulting maximum likelihood estimation problem, it may or may not be computationally feasible to update the parameter estimate $\bar{\theta}$ at each time step. Fortunately, even when it is computationally expensive to estimate $\bar{\theta}$, we can leverage our model confidence as an indicator of when re-estimating these parameters may be most useful. That is, when model confidence degrades that may indicate poor estimates of $\bar{\theta}$.

3.4 Prediction Examples

We illustrate these inference steps with two sets of examples: our running pedestrian example and a simple model of a car.

3.4.1 Pedestrian model (running example)

So far, we have presented a running example of a quadcopter avoiding a human. We use a deliberately simple, purely kinematic model of continuous-time human motion:

$$\dot{x}_H = \begin{bmatrix} \dot{h}_x \\ \dot{h}_y \end{bmatrix} = \begin{bmatrix} v_H \cos u_H \\ v_H \sin u_H \end{bmatrix} . \tag{3.9}$$

However, as discussed in Section 3.2.3, the proposed prediction method operates in discrete time (and space). The discrete dynamics corresponding to (3.9) are given by

$$\begin{aligned} x_H^{k+1} - x_H^k &\equiv x_H(t + \Delta t) - x_H(t) \\ &= \begin{bmatrix} v_H \Delta t \cos u_H(t) \\ v_H \Delta t \sin u_H(t) \end{bmatrix} , \end{aligned} \tag{3.10}$$

for a time discretization of Δt.

3.4.2 Dubins car model

To emphasize the generality of our method, we present similar results for a different application domain: autonomous driving. We will model a human-driven vehicle as a dynamical system whose state x_H evolves as

$$\dot{x}_H = \begin{bmatrix} \dot{h}_x \\ \dot{h}_y \\ \dot{h}_\phi \end{bmatrix} = \begin{bmatrix} v_H \cos h_\phi \\ v_H \sin h_\phi \\ u_H \end{bmatrix} . \tag{3.11}$$

Observe that, while (3.11) appears very similar to (3.9), in this Dubins car example the angle of motion is a *state*, not a *control input*.

We discretize these dynamics by integrating (3.11) from t to $t+\Delta t$, assuming a constant control input u_H:

$$x_H^{k+1} - x_H^k \equiv x_H(t+\Delta t) - x_H(t) = \begin{bmatrix} \frac{v_H}{u_H(t)}\left(\sin\left(h_\phi(t) + u_H(t)\Delta t\right) - \sin(h_\phi(t))\right) \\ -\frac{v_H}{u_H(t)}\left(\cos\left(h_\phi(t) + u_H(t)\Delta t\right) - \cos(h_\phi(t))\right) \\ u_H \Delta t \end{bmatrix} \tag{3.12}$$

For a specific goal position $g = [g_x, g_y]$, the Q-value corresponding to state-action pair (x_H, u_H) and reward function $r_H(x_H, u_H) = -v_H \Delta t$ (until the goal is reached) may be found by solving a shortest path problem offline.

3.4.3 Accurate Model

First, we consider a scenario in which the robot has full knowledge of the human's goal, and the human moves along the shortest path from a start location to this known goal state. Thus, human motion is well-explained by Q_H.

The first row of Fig. 3.2 illustrates the probability distributions our method predicts for the pedestrian's future state at different times. Initially, the predictions generated by our Bayesian confidence-inference approach (right) appear similar to those generated by the low model confidence predictor (left). However, our method rapidly discovers that Q_H is an accurate description of the pedestrian's motion and generates predictions that match the high model confidence predictor (center). The data used in this example was collected by tracking the motion of a real person walking in a motion capture arena. See Section 3.7.1 for further details.

Likewise, the first row of Fig. 3.3 shows similar results for a human-driven Dubins car model (in simulation) at an intersection. Here, traffic laws provide a strong prior on the human's potential goal states. As shown, our method of Bayesian model confidence inference quickly infers the correct goal and learns that the human driver is acting in accordance with its model Q_H. The resulting predictions are substantially similar to the high-β predictor. The data used in this example was simulated by controlling a Dubins car model along a pre-specified trajectory.

3.4.4 Unmodeled Obstacle

Often, robots do not have fully specified models of the environment. Here, we showcase the resilience of our approach to unmodeled obstacles which the human must avoid. In this scenario, the human has the same start and goal as in the accurate model case, except that there is an obstacle along the way. The robot is unaware of this obstacle, however,

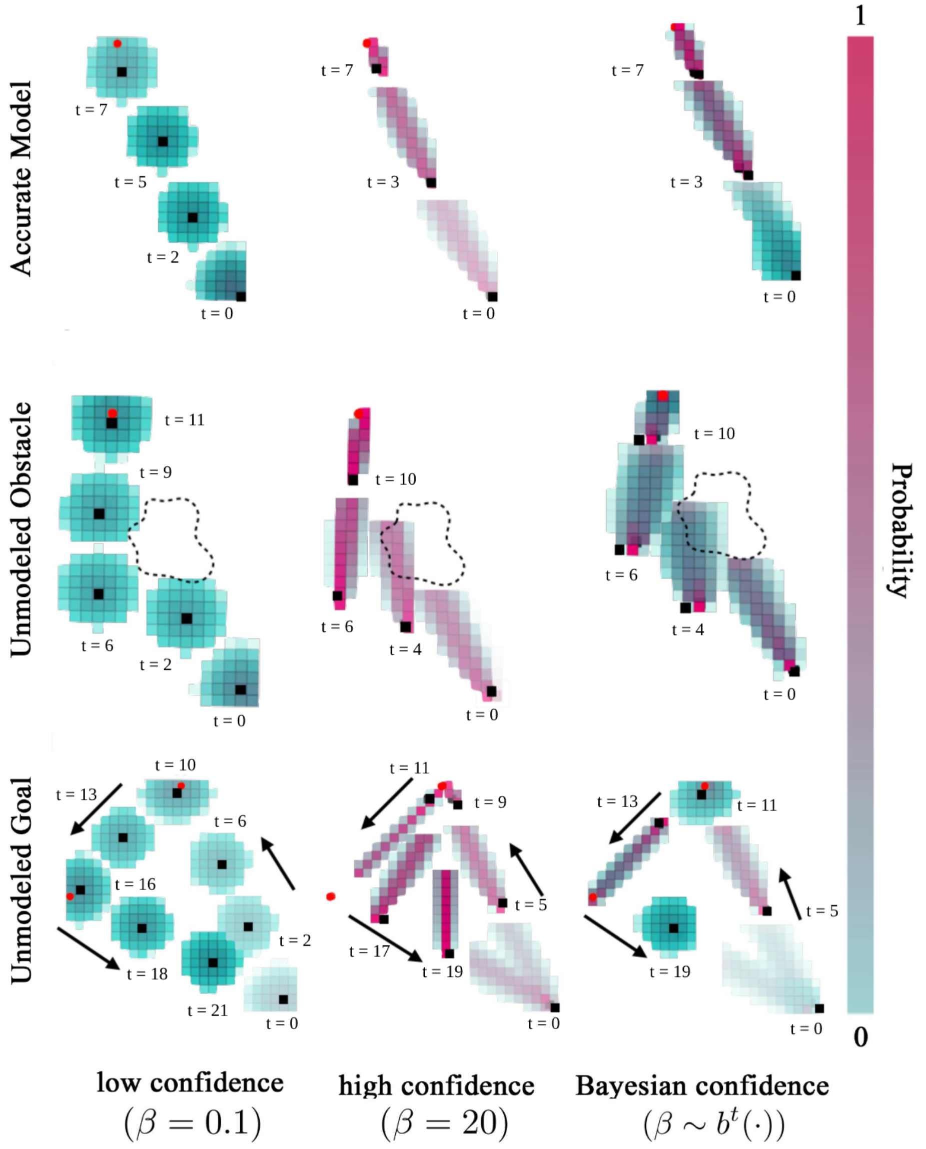


Figure 3.2: Snapshots of pedestrian trajectory and probabilistic model predictions. Top row: Pedestrian moves from the bottom right to a goal marked as a red circle. Middle row: Pedestrian changes course to avoid a spill on the floor. Bottom row: Pedestrian moves to one known goal, then to another, then to a third which the robot has not modeled. The first two columns show predictions for low and high model confidence; the third column shows the predictions using our Bayesian model confidence. For all pedestrian videos, see: https://youtu.be/lh_E9rW-MJo

which means that in its vicinity the human's motion is not well-explained by Q_H, and $b(\beta)$ ought to place more probability mass on higher values of β.

The second rows of Fig. 3.2 and Fig. 3.3 illustrate this type of situation for the pedestrian

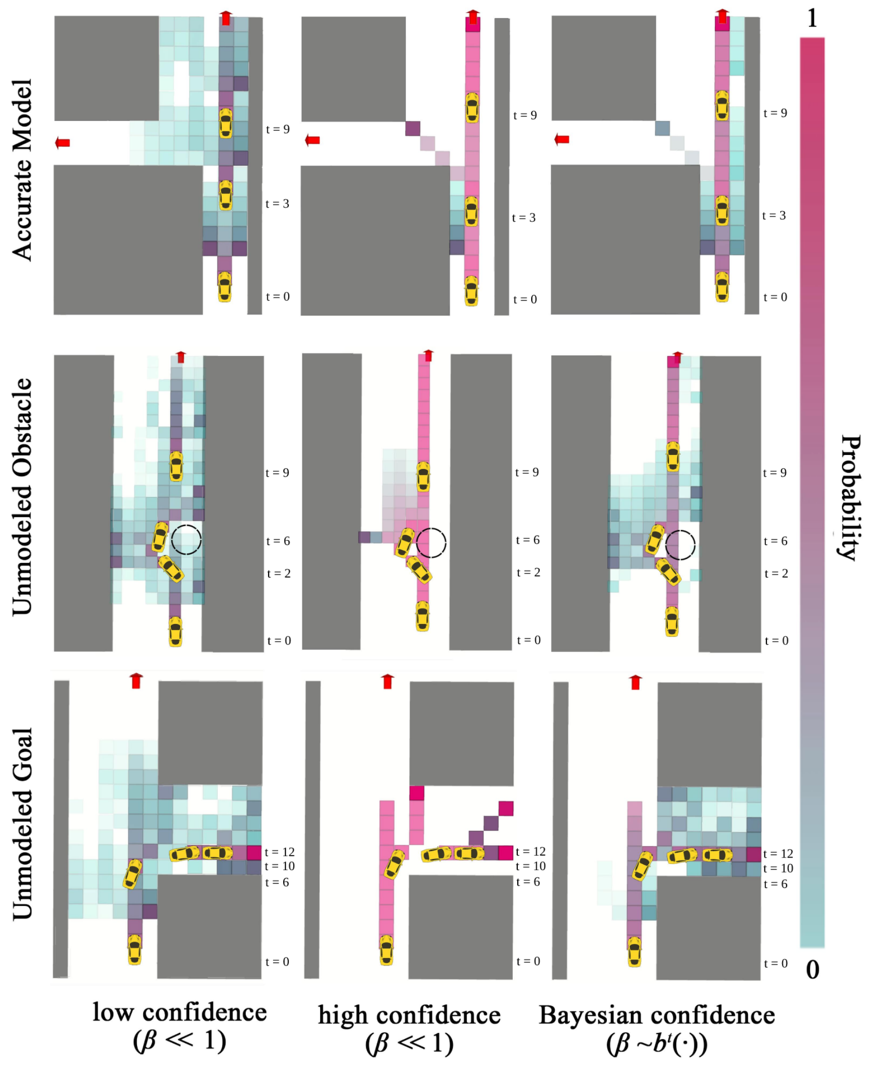

Accurate Model
Unmodeled Obstacle
Unmodeled Goal
t = 0
t = 2
t = 3
t = 6
t = 9
t = 10
t = 12
1
0
Probability
low confidence ($\beta \ll 1$)
high confidence ($\beta \ll 1$)
Bayesian confidence ($\beta \sim b^{t}(\cdot)$)

Figure 3.3: Snapshots of Dubins car and probabilistic predictions. Top row: Car moves straight ahead toward one of two known goals (red arrows), staying in its lane. Middle row: Car suddenly swerves to the left to avoid a pothole. Bottom row: Car turns to the right, away from the only known goal. The left and center columns show results for low and high confidence predictors, respectively, and the right column shows our approach using Bayesian inferred model confidence. For all Dubins car videos, see: https://youtu.be/sAJKNnP42fQ

and Dubins car, respectively. In Fig. 3.2, the pedestrian walks to an *a priori* known goal location and avoids an unmodeled spill on the ground. Analogously, in Fig. 3.3 the car swerves to avoid a large pothole. By inferring model confidence online, our approach generates higher-variance predictions of future state, but only in the vicinity of these unmodeled obstacles. At other times throughout the episode when Q_H is more accurate, our approach produces predictions more in line with the high model confidence predictor.

3.4.5 Unmodeled Goal

In most realistic human-robot encounters, even if the robot does have an accurate environment map and observes all obstacles, it is unlikely for it to be aware of all human goals. We test our approach's resilience to unknown human goals by constructing a scenario in which the human moves between both known and unknown goals.

The third row of Fig. 3.2 illustrates this situation for the pedestrian example. Here, the pedestrian first moves to one known goal position, then to another, and finally back to the start which was not a modelled goal location. The first two legs of this trajectory are consistent with the robot's model of goal-oriented motion, though accurate prediction does require the predictor to infer *which* goal the pedestrian is walking toward. However, when the pedestrian returns to the start, her motion appears inconsistent with Q_H, skewing the robot's belief over β toward zero.

Similarly, in the third row of Fig. 3.3 we consider a situation in which a car makes an unexpected turn onto an unmapped access road. As soon as the driver initiates the turn, our predictor rapidly learns to distrust its internal model Q_H and shift its belief over β upward.

3.5 Safe Probabilistic Planning and Tracking

Given probabilistic predictions of the human's future motion, the robot must plan efficient trajectories which avoid collision with high probability. In order to reason robustly about this probability of future collision, we must account for potential tracking errors incurred by the real system as it follows planned trajectories. To this end, we build on the recent FaSTrack framework of [95], which provides control-theoretic robust safety certificates in the presence of deterministic obstacles, and extend it to achieve approximate probabilistic collision-avoidance.

3.5.1 Background: Fast Planning, Safe Tracking

Recall that x_R is the robot's state for the purposes of motion planning, and that s_R encodes a higher-fidelity, potentially higher-dimensional notion of state (with associated dynamics). The recently proposed FaSTrack framework from [95] uses Hamilton-Jacobi reachability analysis to quantify the worst-case tracking performance of the s_R-system as it follows trajectories generated by the x_R-system. For further reading on reachability analysis refer to [69], [158], and [25]. A byproduct of this FaSTrack analysis is an error feedback controller that the s_R system can use to achieve this worst-case tracking error. The tracking error bound may be given to one of many off-the-shelf real-time motion planning algorithms operating in x_R-space in order to guarantee real-time collision-avoidance by the s_R-system.

Formally, FaSTrack precomputes an optimal tracking controller, as well as a corresponding compact set $\mathcal{E}$ in the robot's planning state space, such that $\big(\pi(s_R(t)) - x_{R,\text{ref}}(t)\big) \in \mathcal{E}$

for *any* reference trajectory proposed by the lower-fidelity planner. This bound $\mathcal{E}$ is a trajectory tracking certificate that can be passed to an online planning algorithm for real-time safety verification: the dynamical robot is guaranteed to *always* be somewhere within the bound relative to the current planned reference point $x_{R,\mathrm{ref}}(t)$. This tracking error bound may sometimes be expressed analytically; otherwise, it may be computed numerically offline using level set methods, e.g. [157]. Equipped with $\mathcal{E}$, the planner can generate safe plans online by ensuring that the entire tracking error bound around the nominal state remains collision-free throughout the trajectory. Efficiently checking these $\mathcal{E}$-augmented trajectories for collisions with known obstacles is critical for real-time performance. Note that the planner only needs to know $\mathcal{E}$ (which is computed offline) and otherwise requires no explicit understanding of the high-fidelity model.

Running example: *Since dynamics (3.2) are decoupled in the three spatial directions, the bound $\mathcal{E}$ computed by FaSTrack is an axis-aligned box of dimensions $\mathcal{E}_x \times \mathcal{E}_y \times \mathcal{E}_z$. For further details refer to [78].*

3.5.2 Robust Tracking, Probabilistic Safety

Unfortunately, planning algorithms for collision checking against deterministic obstacles cannot be readily applied to our problem. Instead, a trajectory's collision check should return the probability that it *might* lead to a collision. Based on this probability, the planning algorithm can discriminate between trajectories that are *sufficiently safe* and those that are not.

As discussed in Section 3.2.4, a safe online motion planner invoked at time t should continually check the probability that, at any future time τ, $(\pi(s_R(\tau)), x_H(\tau)) \in \mathcal{K}$. The tracking error bound guarantee from FaSTrack allows us to conduct worst-case analysis on collisions given a human state x_H. Concretely, if no point in the Minkowski sum $\{x_R + \mathcal{E}\}$ is in the collision set with x_H, we can guarantee that the robot is not in collision with the human.

The probability of a collision event for any point $x_R(\tau)$ along a candidate trajectory is then

$$P_{\mathrm{coll}}\big(x_R(\tau)\big) := P\big((x_R, x_H) \in \mathcal{K}\big) \ . \tag{3.13}$$

Assuming worst-case tracking error bound $\mathcal{E}$, this quantity can be upper-bounded by the total probability that $x_H(\tau)$ will be in collision with *any* of the possible robot states $\tilde{x}_R \in \{x_R(\tau) + \mathcal{E}\}$. For each robot planning state $x_R \in \mathbb{R}^{n_R}$ we define the set of human states in potential collision with the robot:

$$\begin{aligned} \mathcal{H}_{\mathcal{E}}(x_R) := \{\tilde{x}_H \in \mathbb{R}^{n_H} : \\ \exists \tilde{x}_R \in \{x_R + \mathcal{E}\}, (\tilde{x}_R, \tilde{x}_H) \in \mathcal{K}\} \ . \end{aligned} \tag{3.14}$$

Running example: *Given $\mathcal{K}$ and $\mathcal{E}$, $\mathcal{H}_{\mathcal{E}}(x_R)$ is the set of human positions within the rectangle of dimensions $(l + \mathcal{E}_x) \times (l + \mathcal{E}_y)$ centered on $[p_x, p_y]$. A human anywhere in this rectangle could be in collision with the quadcopter.*

The following result follows directly from the definition of the tracking error bound and a union bound.

Proposition 1. *The probability of a robot with worst case tracking error $\mathcal{E}$ colliding with the human at any trajectory point $x_R(\tau)$ is bounded above by the probability mass of $x_H(\tau)$ contained within $\mathcal{H}_\mathcal{E}(x_R(\tau))$.*

We shall consider *discrete-time* motion plans. The probability of collision along any such trajectory from current time step k to final step $k+K$ is upper-bounded by:

$$P_{\text{coll}}^{k:k+K} \leq \overline{P_{\text{coll}}^{k:k+K}} := 1 - \prod_{\kappa=k}^{k+K} P\Big(x_H^\kappa \notin \mathcal{H}_\mathcal{E}(x_R^\kappa) \mid x_H^\kappa \notin \mathcal{H}_\mathcal{E}(x_R^s), k \leq s < \kappa\Big). \tag{3.15}$$

Evaluating the right hand side of (3.15) exactly requires reasoning about the joint distribution of human states over all time steps and its conditional relationship on whether collision has yet occurred. This is equivalent to maintaining a probability distribution over the exponentially large space of trajectories $x_H^{k:k+K}$ that the human might follow. As motion planning occurs in real-time, we shall resort to a heuristic approximation of (3.15).

One approach to approximating (3.15) is to assume that the event $x_H^{\kappa_1} \notin \mathcal{H}_\mathcal{E}(x_R^{\kappa_1})$ is independent of $x_H^{\kappa_2} \notin \mathcal{H}_\mathcal{E}(x_R^{\kappa_2}), \forall \kappa_1 \neq \kappa_2$. This independence assumption is equivalent to removing the conditioning in (3.15). Unfortunately, this approximation is excessively pessimistic; if there is no collision at time step κ, then collision is also unlikely at time step $\kappa+1$ because both human and robot trajectories are continuous. In fact, for sufficiently small time discretization Δt and nonzero collision probabilities at each time step, the total collision probability resulting from an independence assumption would approach 1 exponentially fast in the number of time steps K.

We shall refine this approximation by finding a tight lower bound on the right hand side of (3.15). Because collision events are correlated in time, we first consider replacing each conditional probability $P\big(x_H^\kappa \notin \mathcal{H}_\mathcal{E}(x_R^\kappa) \mid x_H^s \notin \mathcal{H}_\mathcal{E}(x_R^s), k \leq s < \kappa\big)$ by 1 for all $\kappa \in \{k+1, \dots, k+K\}$. This effectively lower bounds $\overline{P_{\text{coll}}^{k:k+K}}$ by the worst case probability of collision at the current time step k:

$$\begin{aligned} \overline{P_{\text{coll}}^{k:k+K}} &\geq 1 - P\big(x_H^k \notin \mathcal{H}_\mathcal{E}(x_R^k)\big) \\ &= P\big(x_H^k \in \mathcal{H}_\mathcal{E}(x_R^k)\big) \ . \end{aligned} \tag{3.16}$$

This bound is extremely loose in general, because it completely ignores the possibility of future collision. However, note that probabilities in the product in (3.15) may be conditioned in any particular order (not necessarily chronological). This commutativity allows us to generate $K-k+1$ lower bounds of the form $\overline{P_{\text{coll}}^{k:k+K}} \geq P\big(x_H^\kappa \in \mathcal{H}_\mathcal{E}(x_R^\kappa)\big)$ for $\kappa \in \{k, \dots, k+K\}$. Taking the tightest of all of these bounds, we can obtain an informative, yet quickly computable, approximator for the sought probability:

$$\overline{P_{\text{coll}}^{k:k+K}} \geq \max_{\kappa \in \{k:k+K\}} P\big(x_H^\kappa \in \mathcal{H}_\mathcal{E}(x_R^\kappa)\big) \approx P_{\text{coll}}^{k:k+K} \tag{3.17}$$

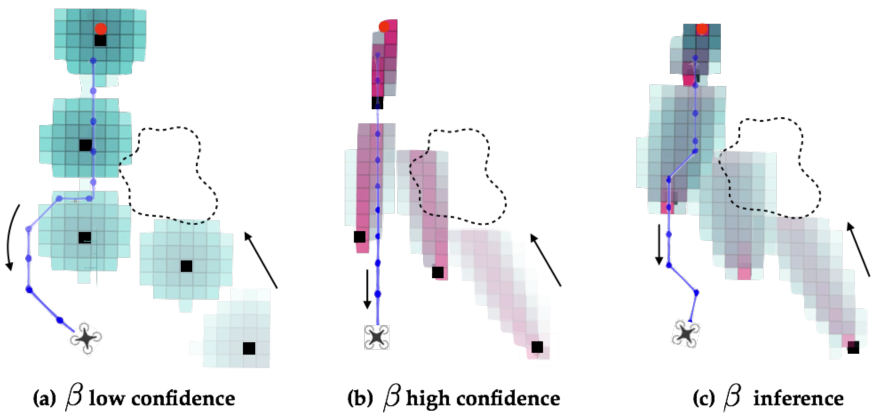


Figure 3.4: Scenario from the middle row of Fig. 3.2 visualized with robot's trajectory. When β is low and the robot is not confident, it makes large deviations from its path to accommodate the human. When β is high, the robot refuses to change course and comes dangerously close to the human. With inferred model confidence, the robot balances safety and efficiency with a slight deviation around the human.

To summarize, the left inequality in (3.17) lower-bounds $\overline{P_{\text{coll}}^{k:k+K}}$ with the greatest marginal collision probability at *any point* in the trajectory. On the right side of (3.17), we take this greatest marginal collision probability as an approximator of the actual probability of collision over the entire trajectory. In effect, we shall approximate $P_{\text{coll}}^{k:k+K}$ with a tight lower bound *of an upper bound*. While this type of approximation may err on the side of optimism, we note that both the robot's ability to replan over time and the fact that the left side of (3.17) is an *upper bound* on total trajectory collision probability mitigate this potentially underestimated risk.

3.5.3 Safe Online Planning under Uncertain Human Predictions

This approximation of collision probability allows the robot to discriminate between valid and invalid candidate trajectories during motion planning. Using the prediction methodology proposed in Section 3.3, we may quickly generate, at every time t, the marginal probabilities in (3.17) at each future time $\kappa \in \{k, \dots, k+K\}$, based on past observations at times $0, \dots, k$. The planner then computes the instantaneous probability of collision $P\big(x_H^\kappa \in \mathcal{H}_\mathcal{E}(x_R^\kappa)\big)$ by integrating $P\big(x_H^\tau \mid x_H^{0:k}\big)$ over $\mathcal{H}_\mathcal{E}(x_R^\kappa)$, and rejects the candidate point x_R^κ if this probability exceeds P_{th}.

Note that for graph-based planners that consider candidate trajectories by generating a graph of time-stamped states, rejecting a candidate edge from this graph is equivalent to rejecting all further trajectories that would contain that edge. This early rejection rule is consistent with the proposed approximation (3.17) of $P_{\text{coll}}^{k:k+K}$ while preventing unnecessary exploration of candidate trajectories that would ultimately be deemed unsafe.

Throughout operation, the robot follows each planned trajectory using the error feedback controller provided by FaSTrack, which ensures that the robot's high-fidelity state representation s_R and the lower-fidelity state used for planning, x_R, differ by no more than the tracking error bound $\mathcal{E}$. This planning and tracking procedure continues until the robot reaches its desired goal state.

Running example: *Our quadcopter is now required to navigate to a target position shown in Fig. 3.4 without colliding with the human. Our proposed algorithm successfully avoids collisions at all times, replanning to leave greater separation from the human whenever her motion departs from the model. In contrast, robot planning with fixed model confidence is either overly conservative at the expense of time and performance or overly aggressive at the expense of safety.*

3.6 Connections to Reachability Analysis

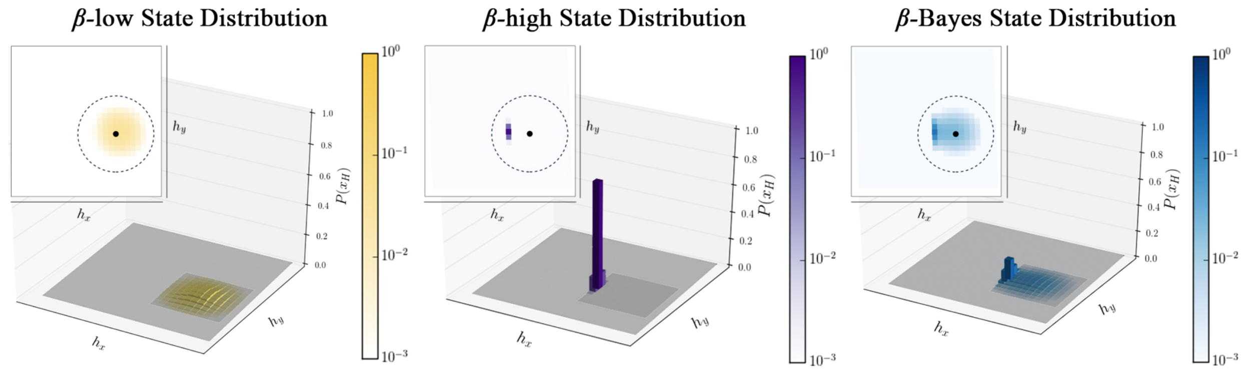


Figure 3.5: The human (black dot) is moving west towards a goal. Visualized are the predicted state distributions for one second into the future when using low, high, and Bayesian model confidence. Higher-saturation indicates higher likelihood of occupancy. The dashed circle represents the pedestrian's 1 second forward reachable set.

In this section, we present an alternative, complementary analysis of the overall safety properties of the proposed approach to prediction and motion planning. This discussion is grounded in the language of reachability theory and worst-case analysis of human motion.

3.6.1 Forward Reachable Set

Throughout this section, we frequently refer to the human's time-indexed forward reachable set. We define this set formally below.

Definition 1. *(Forward Reachable Set) For a dynamical system $\dot{x} = f(x, u)$ with state trajectories given by the function $\xi\big(x(0), t, u(\cdot)\big) =: x(t)$, the forward reachable set* $\mathrm{FRS}(x, t)$ *of a state x after*

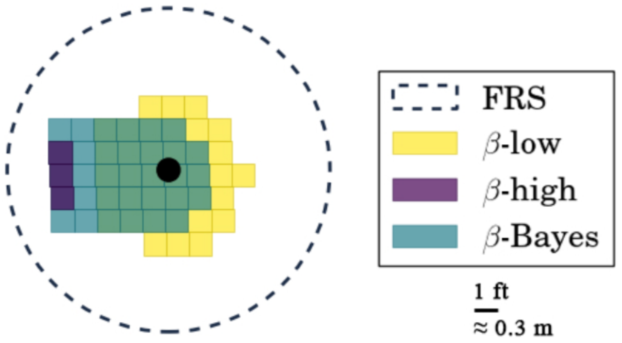


Figure 3.6: Visualization of the states with probability greater than or equal to the collision threshold, $P_{\text{th}} = 0.01$. The human's forward reachable set includes the set of states assigned probability greater than P_{th}. We show these "high probability" predicted states for predictors with fixed low and high β, as well as our Bayesian-inferred β.

time t has elapsed is

$$\text{FRS}(x, t) := \{x' : \exists u(\cdot), x' = \xi(x, t, u(\cdot))\} \,.$$

That is, a state x' is in the forward reachable set of x after time t if it is *reachable* via some applied control signal $u(\cdot)$.

Remark 1. *(Recovery of* FRS*) For $P_{th} = 0$ and any finite β, the set of states assigned probability greater than P_{th} is identical to the forward reachable set, up to discretization errors. This is visualized for low, high, and Bayesian model confidence in Fig.* 3.5.

3.6.2 A Sufficient Condition for the Safety of Individual Trajectories

In Section 3.5.2, we construct an approximation to the probability of collision along a trajectory, which we use during motion planning to avoid potentially dangerous states. To make this guarantee of collision-avoidance for a motion plan even stronger, it would suffice to ensure that the robot never comes too close to the human's forward reachable set. More precisely, a planned trajectory is safe if $\{x_R(t) + \mathcal{E}\} \cap \text{FRS}(x_H, t) = \emptyset$, for every state $x_R(t)$ along a motion plan generated when the human was at state x_H. The proof of this statement follows directly from the properties of the tracking error bound $\mathcal{E}$ described in Section 3.5.

While this condition may seem appealing, it is in fact highly restrictive. The requirement of avoiding the full forward reachable set is not always possible in confined spaces; indeed, this was our original motivation for wanting to predict human motion (see Section 3.2.4). However, despite this shortcoming, the logic behind this sufficient condition for safety provides insight into the effectiveness of our framework.

3.6.3 Recovering the Forward Reachable Set

Though it will not constitute a formal safety guarantee, we analyze the empirical safety properties of our approach by examining how our predicted state distributions over time relate to forward reachable sets. During operation, our belief over model confidence β evolves to match the degree to which the utility model Q_H explains recent human motion. The "time constant" governing the speed of this evolution may be tuned by the system designer to be arbitrarily fast by choosing the parameter ϵ to be small, as discussed in Section 3.3.1. Thus, we may safely assume that $b(\beta)$ places high probability mass on small values of β as soon as the robot observes human motion which is not well explained by Q_H.

Fig. 3.6 shows the sets of states with "high enough" ($> P_{\text{th}}$) predicted probability mass overlaid on the human's forward reachable set at time t, which is a circle of radius $v_H t$ centered on x_H for the dynamics in our running example. When β is high (10), we observe that virtually all of the probability mass is concentrated in a small number of states in the direction of motion predicted by our utility model. When β is low (0.05) we observe that the set of states assigned probability above our collision threshold P_{th} occupies a much larger fraction of the reachable set. A typical belief $b(\beta)$ recorded at a moment when the human was roughly moving according to Q_H yields an intermediate set of states.

Fig. 3.7 illustrates the evolution of these sets of states over time, for the unmodeled obstacle example of Section 3.7.1 in which a pedestrian avoids a spill. Each row corresponds to the predicted state distribution at a particular point in time. Within a row, each column shows the reachable set and the set of states assigned occupancy probability greater than $P_{\text{th}} = 0.01$. The color of each set of states corresponds to the value of β used by the low confidence and high confidence predictors, and the *maximum a posteriori* value of β for the Bayesian confidence predictor. The human's known goal state is marked by a red dot.

Interestingly, as the Bayesian model confidence decreases—which occurs when the pedestrian turns to avoid the spill at $t \approx 6$ s—the predicted state distribution assigns high probability to a relatively large set of states, but unlike the low-β predictor that set of states is oriented toward the known goal. Of course, had $b(\beta)$ placed even more probability mass on lower values of β then the Bayesian confidence predictor would converge to the low confidence one.

Additionally, we observe that, within each row as the prediction horizon increases, the area contained within the forward reachable set increases and the fraction of that area contained within the predicted sets decreases. This phenomenon is a direct consequence of our choice of threshold P_{th}. Had we chosen a smaller threshold value, a larger fraction of the forward reachable set would have been occupied by the lower-β predictors.

This observation may be viewed prescriptively. Recalling the sufficient condition for safety of planned trajectories from Section 3.6.2, if the robot replans every T_{replan} seconds, we may interpret the fraction of $\text{FRS}(\cdot, t + T_{\text{replan}})$ assigned occupancy probability greater than P_{th} by the low confidence predictor as a rough indicator of the safety of an individual motion plan, robust to worst-case human movement. As this fraction tends toward unity,

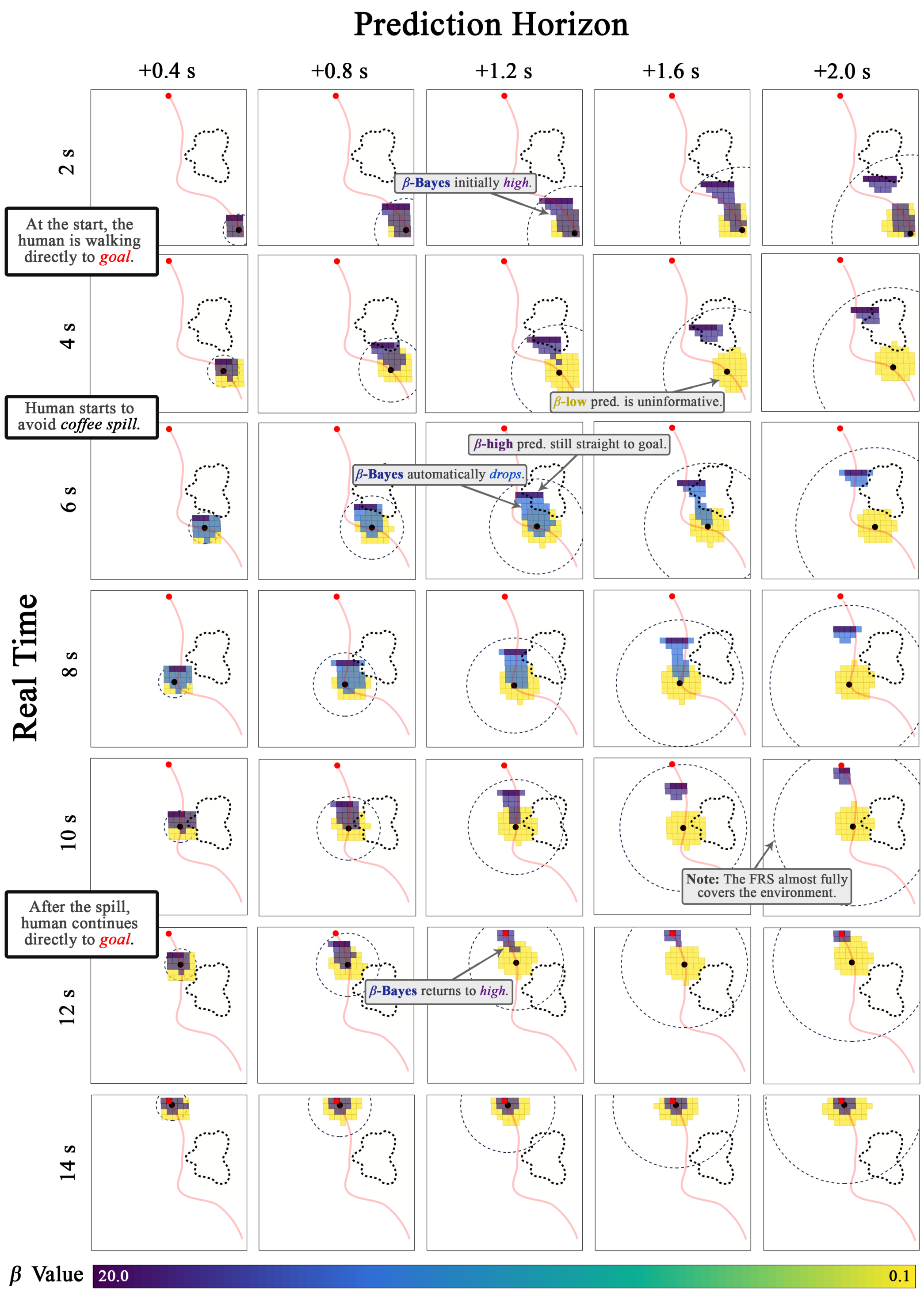


Figure 3.7: The human (black dot) is walking towards the known goal (red dot) but has to avoid an unmodeled coffee spill on the ground. Here we show the snapshots of the predictions at various future times (columns) as the human walks around in real time (rows). The visualized states have probability greater than or equal to $P_{\text{th}} = 0.01$. Each panel displays the human prediction under low confidence (in yellow), high confidence (in dark purple), and Bayesian confidence (colored as per the most likely β value), as well as the forward reachable set. The human's actual trajectory is shown in red.

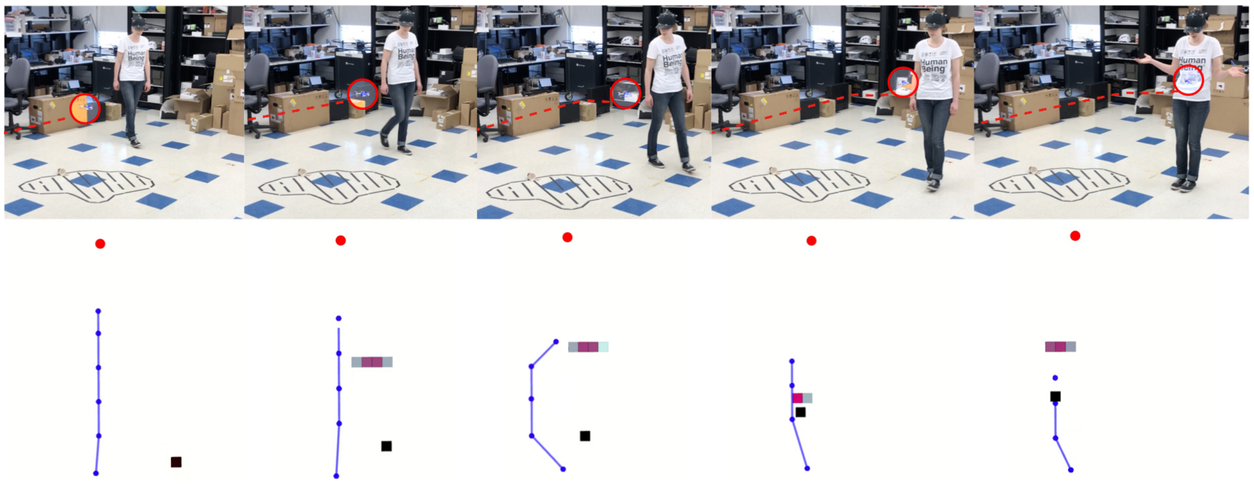


Figure 3.8: Predicting with fixed-β (in this case, $\beta = 20$) can yield highly inaccurate predictions (and worse, confidently inaccurate ones). The subsequent motion plans may not be safe; here, poor prediction quality leads to a collision.

the robot is more and more likely to be safe. However, for any $P_{\text{th}} > 0$, this fraction approaches zero for $T_{\text{replan}} \uparrow \infty$. This immediately suggests that, if we wish to replan every T_{replan} seconds, we can achieve a particular level of safety as measured by this fraction by choosing an appropriate threshold P_{th}.

In summary, confidence-aware predictions rapidly place high probability mass on low values of β whenever human motion is not well-explained by utility model Q_H. Whenever this happens, the resulting predictions encompass a larger fraction of the forward reachable set, and in the limit that $P_{\text{th}} \downarrow 0$ we recover the forward reachable set exactly. The larger this fraction, the more closely our approach satisfies the sufficient condition for safety presented in Section 3.6.2.

3.7 Hardware Demonstration with Real Humans

We implemented confidence-aware human motion prediction (Section 3.3) and integrated it into a real-time, safe probabilistic motion planner (Section 3.5), all within the Robot Operating System (ROS) software framework of [169]. To demonstrate the efficacy of our methods, we tested our work for the quadcopter-avoiding-pedestrian example used for illustration throughout this paper. Human trajectories were recorded as (x, y) positions on the ground plane at roughly 235 Hz by an OptiTrack infrared motion capture system, and we used a Crazyflie 2.0 micro-quadcopter, also tracked by the OptiTrack system.[2]

[2]We note that in a more realistic setting, we would require alternative methods for state estimation using other sensors, such as lidar and/or camera(s). A video recording may be found at https://youtu.be/2ZRGxWknENg.

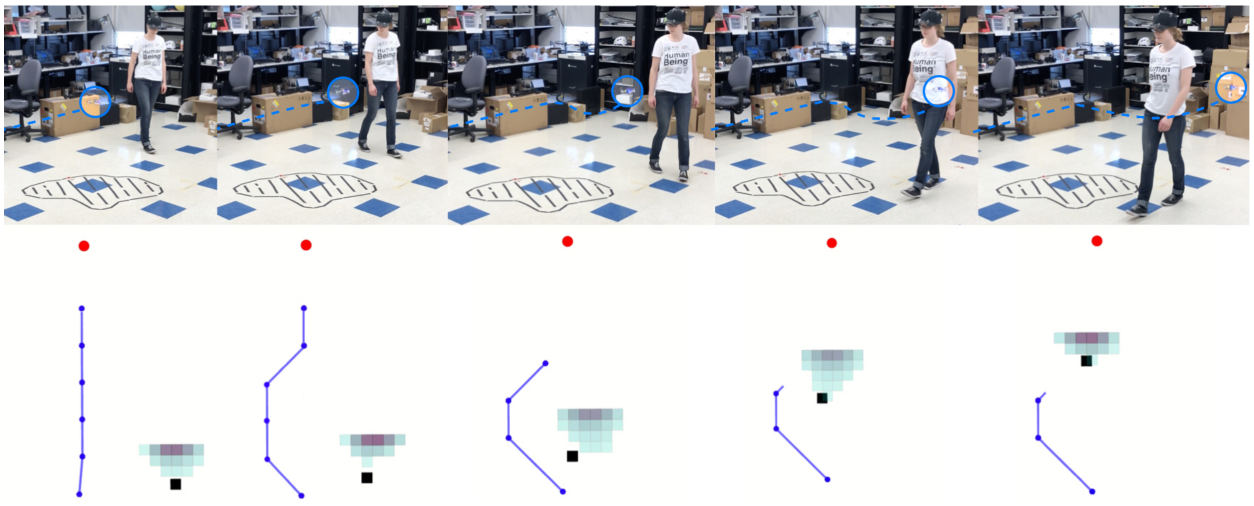


Figure 3.9: Inferring β leads to predicted state distributions whose entropy increases whenever the utility model Q_H fails to explain observed human motion. The resulting predictions are more robust to modeling errors, resulting in safer motion plans. Here, the quadcopter successfully avoids the pedestrian even when she turns unexpectedly.

Fig. 3.4 illustrates the unmodeled obstacle case from Section 3.7.1, in which the pedestrian turns to avoid a spill on the ground. Using a low model confidence results in motion plans that suddenly and excessively deviate from the ideal straight-line path when the pedestrian turns to avoid the spill. By contrast, the high confidence predictor consistently predicts that the pedestrian will walk in a straight line to the goal even when they turn; this almost leads to collision, as shown detail in Fig. 3.8. Our proposed approach for Bayesian model confidence initially assigns high confidence and predicts that the pedestrian will walk straight to the goal, but when they turn to avoid the spill, the predictions become less confident. This causes the quadcopter to make a minor course correction, shown in further detail in Fig 3.9.

3.7.1 Evaluation with Real Human Trajectories

To demonstrate the characteristic behavior of our approach, we created three different environment setups and collected a total of 48 human walking trajectories (walked by 16 different people). The trajectories are measured as (x, y) positions on the ground plane at roughly 235 Hz by an OptiTrack infrared motion capture system.[3]

Environments. In the first environment there are no obstacles and the robot is aware of the human's goal. The second environment is identical to the first, except that the human

[3]We note that in a more realistic setting, we would need to utilize alternative methods for state estimation such as lidar measurements.

must avoid a coffee spill that the robot is unaware of. In the third environment, the human walks in a triangular pattern from her start position to two known goals and back.

Evaluated Methods. For each human trajectory, we compare the performance of our adaptive β inference method with two baselines using fixed $\beta \in \{0.05, 10\}$. When $\beta = 0.05$, the robot is unsure of its model of the human's motion. This low-confidence method cannot trust its own predictions about the human's future trajectory. On the other hand, the $\beta = 10$ high-confidence method remains confident in its predictions even when the human deviates from them. These two baselines exist at opposite ends of a spectrum. Comparing our adaptive inference method to these baselines provides useful intuition for the relative performance of all three methods in common failure modes (see Fig. 3.4).

Metrics. We measure the performance of our adaptive β inference approach in both of these cases by simulating a quadcopter moving through the environment to a pre-specified goal position while replaying the recorded human trajectory. We simulate near-hover quadcopter dynamics with the FaSTrack optimal controller applied at 100 Hz. For each simulation, we record the minimum distance in the ground plane between the human and the quadcopter as a proxy for the overall safety of the system. The quadcopter's travel time serves to measure its overall efficiency.

In each environment, we compute the safety metric for all 16 human trajectories when applying each of the three human motion prediction methods and display the corresponding box and whisker plots side by side. To compare the efficiency of our approach to the baselines we compute the difference between the trajectory completion time of our approach, T_{infer}, and that of the low and high confidence baselines, $\{T_{\text{lo}}, T_{\text{hi}}\}$. If the resulting boxplots are below zero, then β inference results in faster robot trajectories than the baselines on a per-human trajectory basis.[4]

Complete Model. First, we designed an example environment where the robot's model is complete and the human motion appears to be rational. In this scenario, humans would walk in a straight line from their start location to their goal which was known by the robot *a priori*.

When the robot has high confidence in its model, the human's direct motion towards the goal appears highly rational and results in both safe (Fig. 3.10, top left) and efficient plans (Fig. 3.10, bottom left). We see a similar behavior for the robot that adapts its confidence: although initially the robot is uncertain about how well the human's motion matches its model, the direct behavior of the human leads to the robot to believe that it has high model confidence. Thus, the β inference robot produces overall safe and efficient plans. Although we expect that the low-confidence model would lead to less efficient plans but comparably safe plans, we see that the low-confidence robot performs comparably in terms of both safety and efficiency.

Ultimately, this example demonstrates that when the robot's model is rich enough to

[4]The upper and lower bounds of the box in each boxplot are the 75th and 25th percentiles. The horizontal red line is the median, and the notches show the bootstrapped 95% confidence interval for the population mean.

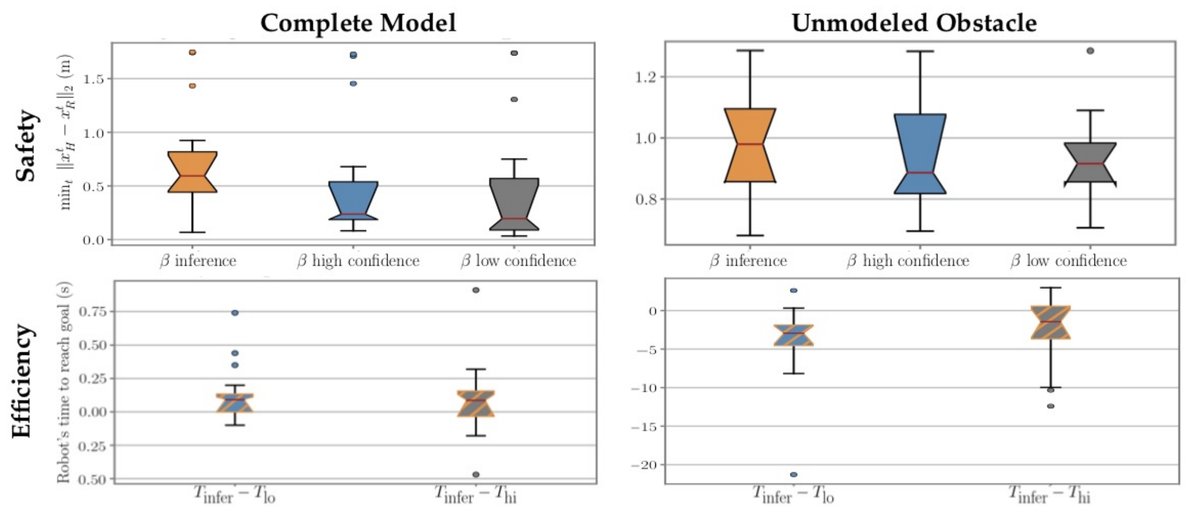


Figure 3.10: Safety and efficiency metrics in a complete environment and one with an unmodeled obstacle.

capture the environment and behavior of the human, inferring model confidence does not hinder the robot from producing safe and efficient plans.

Unmodeled Obstacle. Often, robots do not have fully specified models of the environment. In this scenario, the human has the same start and goal as in the complete model case except that there is a coffee spill in her path. This coffee spill on the ground is unmodeled by the robot, making the human's motion appear less rational.

When the human is navigating around the unmodeled coffee spill, the robot that continuously updates its model confidence and replans with the updated predictions almost always maintains a safe distance (Fig. 3.10, top right). In comparison, the fixed-β models that have either high-confidence or low-confidence approach the human more closely. This increase in the minimum distance between the human and the robot during execution time indicates that continuous β inference can lead to safer robot plans.

For the efficiency metric, a robot that uses β inference is able to get to the goal faster than a robot that assumes a high or a low confidence in its human model (Fig. 3.10, bottom right). This is particularly interesting as overall we see that enabling the robot to reason about its model confidence can lead to *safer* and *more efficient* plans.

Unmodeled Goal. In most realistic human-robot encounters, even if the robot does have an accurate environment map and observes all obstacles, it is unlikely for it to be aware of all human goals. We test our approach's resilience to unknown human goals by constructing a scenario in which the human moves between both known and unknown goals. The human first moves to two known goal positions, then back to the start. The first two legs of this trajectory are consistent with the robot's model of goal-oriented motion. However, when the human returns to the start, she appears irrational to the robot. Fig. 3.11 and 3.12 summarize the performance of the inferred-β, high-confidence, and low-confidence methods in this scenario. All three methods perform similarly with respect to the minimum distance safety metric in Fig. 3.11. However, Fig. 3.12 suggests

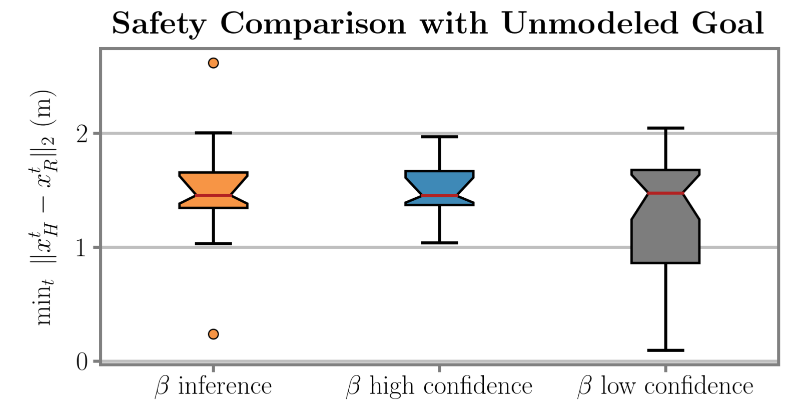


Figure 3.11: Safety results for the unmodeled goal scenario.

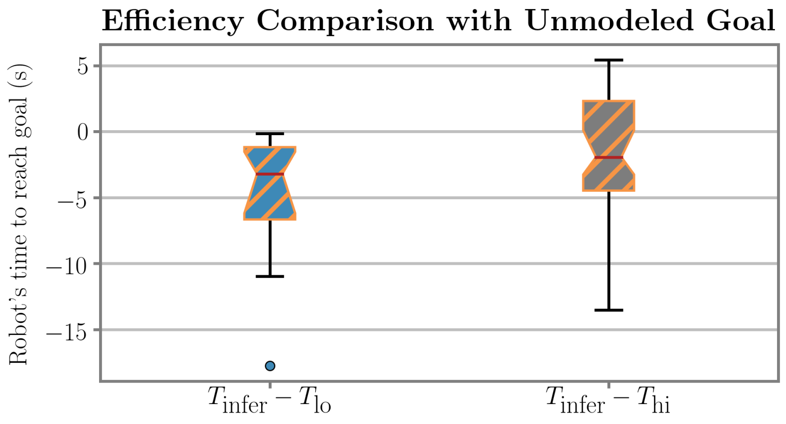


Figure 3.12: Efficiency results for the unmodeled goal scenario.

that the inferred-β method is several seconds faster than both fixed-β approaches. This indicates that, without sacrificing safety, our inferred-β approach allows the safe motion planner to find more efficient robot trajectories.

3.8 Conclusion

When robots operate in complex environments in concert with other agents, safety often depends upon the robot's ability to predict the agents' future actions. While this prediction problem may be tractable in some cases, it can be extremely difficult for agents like people who act with *intent*. In this paper, we introduce the idea of *confidence-aware* prediction as a natural coping mechanism for predicting the future actions of intent-driven agents. Our approach uses each measurement of the human's state to reason about the accuracy of its internal model of human decision making. This reasoning about model confidence is expressed compactly as a Bayesian filter over the possible values of a single parameter, β, which controls the entropy of the robot's model of the human's choice of action. In effect, whenever the human's motion is not well-explained by this model, the robot predicts that the human could occupy a larger volume of the state space.

We couple this notion of confidence-aware prediction with a reachability-based robust motion planning algorithm, FaSTrack, which quantifies the robot's ability to track a planned reference trajectory. Using this maximum tracking error allows us to bound an approximation of the probability of collision along planned trajectories. Additionally, we present a deeper connection between confidence-aware prediction and forward reachable sets, which provides an alternative explanation of the safety of our approach. We demonstrate the proposed methodology on a ROS-based quadcopter testbed in a motion capture arena.

3.8.1 Limitations

There are several important limitations of this work, which we discuss below.

State Discretization. As presented, our approach to prediction requires a discrete representation of the human's state space. This can be tractable for the relatively simple dynamical models of human motion we consider in this work. Fortunately, one of the strongest attributes of confidence-aware prediction is that it affords a certain degree of robustness to modeling errors by design. Still, our approach is effectively limited to low-order dynamical models.

FaSTrack Complexity. FaSTrack provides a strong safety guarantee vis-à-vis the maximum tracking error that could ever exist between a higher-fidelity dynamical model of the robot and a lower-order one used for motion planning. Unfortunately, the computational complexity of finding this maximum tracking error and the corresponding safety controller scales exponentially with the dimension of the high-fidelity model. In some cases, these dynamics are decomposable and analytic solutions exist, e.g. [78, 50], and in other cases conservative approximations may be effective, e.g. [47, 179].

Boltzmann Distributional Assumption. We model the human's choice of control input at each time step as an independent, random draw from a Boltzmann distribution (3.3). This distributional assumption is motivated from the literature in cognitive science and econometrics and is increasingly common in robotics, yet it may not be accurate in all cases. Maintaining an up-to-date model confidence belief $b(\beta)$ can certainly mitigate this inaccuracy, but only at the cost of making excessively conservative predictions.

Safety Certification. Our analysis in Section 3.6 makes connections to forward reachability in an effort to understand the safety properties of our system. As shown, whenever our confidence-aware prediction method detects poor model performance it quickly yields predictions that approximate the human's forward reachable set. Although this approximation is not perfect, and hence we cannot provide a strong safety certificate, the connection to reachability is in some sense prescriptive. That is, it can be used to guide the choice of collision probability threshold P_{th} and replanning frequency. However, even if we could provide a strong guarantee of collision-avoidance for a *particular* motion plan, that would not, in general, guarantee that future motion plans would be *recursively safe*. This recursive property is much more general and, unsurprisingly, more difficult to satisfy.

3.8.2 Future Directions

Future work will aim to address each of these shortcomings. We are also interested in extending our methodology for the multi-robot, multi-human setting; our preliminary results are reported in [15]. Additionally, we believe that our model confidence inference approach could be integrated with other commonly-used probabilistic prediction methods besides the Boltzmann utility model. Finally, we are excited to test our work in hardware in other application spaces, such as manipulation and driving.

Chapter 4

Confidence-aware Game-theoretic Human Models

This chapter is based on the paper "Safety Assurances for Human-Robot Interaction via Confidence-aware Game-theoretic Human Models" [211] written in collaboration with Ran Tian, Liting Sun, Masayoshi Tomizuka, and Anca Dragan.

In this section we focus on maintaining safety in highly dynamic human-robot interactions, such as when an autonomous car merges into a roundabout with an oncoming human-driven vehicle (Fig. 4.1). While planning approaches incorporate safety constraints in diverse ways [193], *safety monitors* have emerged as a desirable additional layer of safety. These methods allow the planner to guide the robot, but compute when imminent collisions would happen and take over control to steer the robot away from danger.

Crucial to these safety monitors is a method for detecting imminent collisions. Typically, this is based on *worst-case* reasoning. A predominant approach, *backwards reachability analysis* [158, 134], treats the human-robot interaction as a zero-sum collision-avoidance game, protecting the robot against *any* controls the human might execute. This leads to safety monitors that do maintain safety, but inhibit the robot's ability to make progress by intervening excessively.

We thus seek a way of making safety monitors less conservative, while still being effective at their primary job—maintaining safety. What makes this challenging is that the moment the zero-sum game assumption is replaced with any human behavior model, the model might be wrong, leading to loss of safety. Our idea is to mediate this issue in two ways: 1) still use a zero-sum collision avoidance game, but instead of allowing the human *any* controls, we use a human *model* to *restrict* set of controls we safeguard against, eliminating those that the model deems very improbable; and 2) detect online *how well the model explains the human*, and use this to adapt the restriction; at the extreme, when the model is completely wrong, our monitor should go back to protecting against any human controls.

Two questions still remain: what human model to use, and how to detect when it is

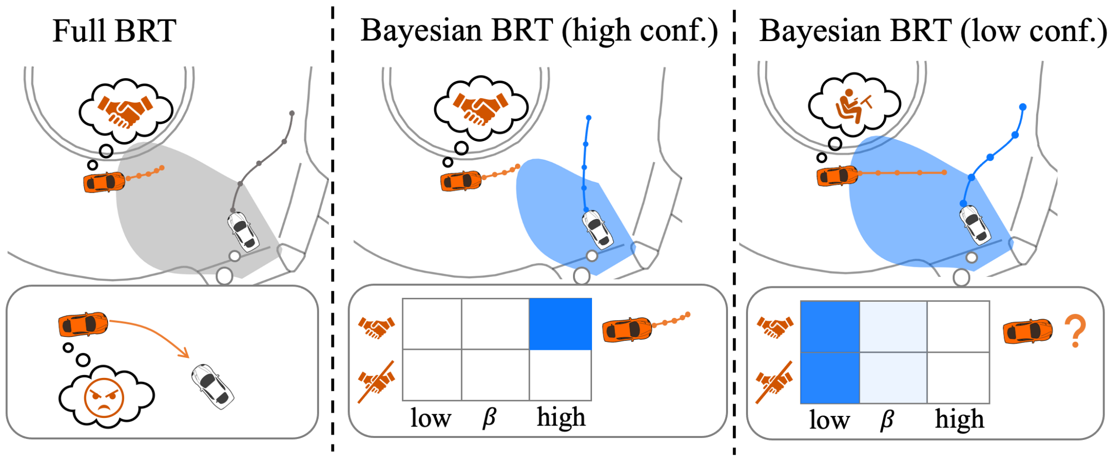


Figure 4.1: Robot car (white) merges into a round-about with a nearby human-driven car (orange). (left) Human accommodates for robot, but robot is overly conservative and protects against the full backwards reachable tube (BRT). (center) Our Bayesian BRT infers how the human is influenced by the robot and shrinks the set of unsafe states. (right) When the human does not behave according to the model, the robot detects this and automatically reverts to the full BRT.

wrong. While models that treat the human as acting in isolation and ignoring the robot are popular [238, 112, 182], they are still very conservative: if the planner tries to merge in front of the human, the safety monitor based on these "human-in-isolation" models would intervene to prevent it, because it has no confidence in the human reacting to the robot and making space—also known as the "frozen robot" problem [214]. For this reason, prior work in planning has introduced models based on *general-sum* games between the human and robot, which account for the human's influence on the robot, but also for the robot's influence on the human [187, 164, 165, 194, 73, 210]. We propose to use such models in safety monitoring too, as a way to restrict the set of controls that the robot safeguards against. In our method, the robot performs backwards reachability analysis but does not worry about human controls that are outside of the bounds of what the general sum game deems likely.

While this reduces the conservatism, no model is perfect and relying solely on the human model might remove controls that the human actually ends up executing. To detect if the model's predictive performance is degrading and increase conservatism when it does, our approach uses online observations of human behavior to assess the quality of the model and adapt the control bound restriction. Building on prior work in safe planning [74], we achieve automatic adaptation by treating human behavior as observations of the human's *rationality* level in the general-sum game. When the human starts executing less-probable controls, the model treats this behavior as more noisy, and deems more future human controls to be likely to occur than before. In turn, this propagates to a larger human control bound in our backwards reachability analysis and a more conservative safety monitor.

We test our approach in simulated interactions, as well as on real human driving data. Our results suggest that it can effectively enable robots to modulate the conservatism of their safety monitors, ultimately leading to more efficient (i.e., higher reward) behaviors that still maintain safety.

4.1 Related Work

Safety for robots operating around humans. Forward reachability methods have been used to compute the set of states where other agents could be in the future [7, 143], after which the robot plans to avoid this set. While safe, these methods often lead to overly-conservative robot behaviors especially in close-proximity interactive scenarios. Prior work used empirical [67] and "human-in-isolation" models [14] to obtain restrictions on human controls and, ultimately, their forward reachable set. However, these methods do not inherently account for the robot's ability to take safety maneuvers in response to the human behavior, introducing additional conservatism. In contrast, backwards reachability methods are grounded in zero-sum dynamic games [158, 134] which encode the robot's ability to enact safety controls. While generally less conservative than full forward reachability, this approach still often suffers from falsely flagging safe states as unsafe because of the full control authority and adversarial nature of the human model. Recent work has attempted to reduce this conservatism by restricting the set of human controls during safety analysis through data-driven human trajectory forecasts [196, 138]. However, this approach blindly trusts the data-driven forecasts and cannot detect model errors; when the quality of the data-driven forecasts degrade, so do the predicted set of human controls, ultimately compromising the safety monitor.

Structured human decision-making models. "Human-in-isolation" models—whereby the human is treated as behaving independently of how other agents nearby behave— have been widely studied and applied in the navigation domain [238, 112, 182]. Recent work has developed model confidence monitors for this class of models [74, 34], enabling robots to detect if and when their human models are misspecified. Unlike human-in-isolation models, theory of mind models capture how humans account for the behavior of others when choosing their actions [20, 41, 81, 202, 209]. Recent work has shown the effectiveness of using such models—specifically, general-sum Stackelberg games [218]—in the context of autonomous driving[187, 194, 73]. In our work, we leverage such game-theoretic human models and introduce a confidence-aware monitor for this modelling paradigm.

4.2 Background

Hamilton-Jacobi reachability. Our method is based on Hamilton-Jacobi (HJ) reachability analysis [158, 155], a mathematical formalism for quantifying the performance and safety of multi-agent dynamical systems. It has been successfully utilized in a range of safety-critical applications such as multi-vehicle planning [51, 61], multi-player reach-avoid games [98, 155], and autonomous driving under occlusions [231] due to its ability to handle general nonlinear dynamics, flexibility to represent unsafe sets of arbitrary shapes, and ability to synthesize safety-preserving robot controllers.

In this work, we use reachability analysis to compute a *backward reachable tube* (BRT), $\mathcal{V}(\tau)$, given an unsafe set of states $\mathcal{L}$ (e.g., all states where the human and robot are in collision). Intuitively, $\mathcal{V}(\tau)$ is the set of states from which if system trajectories start, they are guaranteed to enter into the unsafe set of states within a time horizon of τ despite the robot's best effort to avoid the unsafe set.

Let the dynamics of the human-robot system evolve via $\dot{x} = f(x, u_{\mathrm{H}}, u_{\mathrm{R}})$ where f is assumed to be uniformly continuous in time and Lipschitz continuous in x for fixed u_{H} and u_{R}. Here, $x := [x_{\mathrm{H}}, x_{\mathrm{R}}]^\top \in \mathcal{X}$ is the joint state[1] of the human and robot, and $u_{\mathrm{H}} \in \mathcal{U}_{\mathrm{H}}$ and $u_{\mathrm{R}} \in \mathcal{U}_{\mathrm{R}}$ are the human's and robot's inputs, respectively. We assume that the state of both agents can be accurately sensed at all times [2].

To ensure robustness to the possible—including worst-case—behaviors of the human agent, the computation of the BRT is formulated as a zero-sum differential game between the robot and human. The optimal value of this game can be obtained by solving the final-value Hamilton-Jacobi-Isaacs Variational-Inequality (HJI-VI) via dynamic programming:

$$\begin{aligned} \min \left\{ \frac{\partial V(x,\tau)}{\partial \tau} + H\big(x, \tau, \nabla V(x,\tau)\big), \ell(x) - V(x,\tau) \right\} &= 0, \\ V(x,0) = \ell(x),\ \tau \in [-T, 0], & \end{aligned} \tag{4.1}$$

where $\nabla V(x,\tau)$ is the spatial derivative of the value function and $\ell(x)$ is the implicit surface function encoding the set of unsafe states: $\mathcal{L} = \{x : \ell(x) \leq 0\}$. The Hamiltonian H encodes the effect of the dynamics, robot, and human control on the resulting value and is defined as:

$$H(x, \tau, \nabla V(x,\tau)) = \max_{u_{\mathrm{R}} \in \mathcal{U}_{\mathrm{R}}} \min_{u_{\mathrm{H}} \in \mathcal{U}_{\mathrm{H}}} \nabla V(x,\tau)^\top f(x, u_{\mathrm{H}}, u_{\mathrm{R}}). \tag{4.2}$$

After computing the value function $V(x,\tau)$ backwards in time over $\tau \in [-T, 0]$, we can obtain the BRT at any time τ by looking at the sub-zero level set of the value function:

$$\mathcal{V}(\tau) := \{x : V(x,\tau) \leq 0\}. \tag{4.3}$$

This encodes the set of initial (joint) states from which there does not exist a dynamically-feasible safety control for the robot to perform to avoid the human. HJ reachability also synthesizes the robot's optimal safety-preserving control:

$$u_{\mathrm{R}}^*(x,\tau) = \arg\max_{u_{\mathrm{R}} \in \mathcal{U}_{\mathrm{R}}} \min_{u_{\mathrm{H}} \in \mathcal{U}_{\mathrm{H}}} \nabla V(x,\tau)^\top f(x, u_{\mathrm{H}}, u_{\mathrm{R}}), \tag{4.4}$$

[1]Other state-space representations can be used. In Section 4.4 we use the relative state between the human and robot to reduce state dimensionality.

[2]State estimation error can be accounted for by incorporating additional margin proportional to the estimation uncertainty into the BRT computation.

which can be used in a least-restrictive fashion by only being applied at the boundary of the unsafe set [25].

Note that in (4.2), the agents are traditionally modelled as optimizing with respect to *all* of their dynamically-feasible control inputs, $\mathcal{U}_\mathrm{i}, i \in \{\mathrm{H}, \mathrm{R}\}$, resulting in an overly conservative BRT. In this work, we aim to reduce the conservatism (i.e., size of $\mathcal{V}(\tau)$) of this safety monitor by detecting emergent leader-follower roles in human-robot interaction and restricting the human's controls $\mathcal{U}_\mathrm{H}$ accordingly.

General-sum Stackelberg games. We model humans as acting according to a discrete-time, general-sum Stackelberg game [218] with the robot wherein each agent takes on the role of being either a "leader" or a "follower". A "leader" maximizes their reward over time subject to the "follower" who must plan their trajectory in response. Intuitively, the leader aims to influence the follower and the follower tends to accommodate the leader. This modelling paradigm is well-suited to capture dynamic interactions such as merging or lane-changing and, unlike zero-sum games, can encode unique high-level objectives for each agent.

Let the human's discrete-time control trajectory over the time horizon T be denoted by $\mathbf{u}_\mathrm{H} = [u_\mathrm{H}^0, \dots, u_\mathrm{H}^T]^\top$ and the robot's discrete-time control trajectory be $\mathbf{u}_\mathrm{R} = [u_\mathrm{R}^0, \dots, u_\mathrm{R}^T]^\top$. Instead of assuming the human is a perfectly rational player, we model them a noisily-optimal player; this is well-suited for human models obtained via inverse reinforcement learning [237, 221, 135] and naturally accounts for model inaccuracies and noisy human behavior. In an open-loop Stackelberg game, a noisily-optimal *follower* human chooses their control trajectory from a distribution conditioned on the leader robot's control trajectory:

$$P(\mathbf{u}_\mathrm{H} \mid x^0, \mathbf{u}_\mathrm{R}; \beta) = \frac{1}{Z_1} e^{\beta R_\mathrm{H}(x^0, \mathbf{u}_\mathrm{H}, \mathbf{u}_\mathrm{R})}, \tag{4.5}$$

where $Z_1 := \int_{\mathbf{u}_\mathrm{H}} e^{\beta R_\mathrm{H}(x^0, \mathbf{u}_\mathrm{H}, \mathbf{u}_\mathrm{R})} d\mathbf{u}_\mathrm{H}$ is the partition function when the human is a follower and the human's cumulative reward is $R_\mathrm{H}(x^0, \mathbf{u}_\mathrm{H}, \mathbf{u}_\mathrm{R}) := \sum_{\tau=0}^{T} r_\mathrm{H}(x^\tau, u_\mathrm{H}^\tau, u_\mathrm{R}^\tau)$ where $r_\mathrm{H}(\cdot, \cdot, \cdot)$ is the instantaneous reward.

However, in reality, the human could assume the role of either a leader or follower during an interaction with the robot. Specifically, a human who assumes the role of a *leader* will draw their control trajectory from the distribution:

$$P(\mathbf{u}_\mathrm{H} \mid x^0; \beta) = \frac{1}{Z_2} e^{\beta R_\mathrm{H}(x^0, \mathbf{u}_\mathrm{H}, \mathbf{u}_\mathrm{R}^*(x^0, \mathbf{u}_\mathrm{H}))}, \tag{4.6}$$

where $Z_2 := \int_{\mathbf{u}_\mathrm{H}} e^{\beta R_\mathrm{H}(x^0, \mathbf{u}_\mathrm{H}, \mathbf{u}_\mathrm{R}^*(x^0, \mathbf{u}_\mathrm{H}))} d\mathbf{u}_\mathrm{H}$ is the partition function and $\mathbf{u}_\mathrm{R}^*(x^0, \mathbf{u}_\mathrm{H})$ is the robot follower's best response to the human's control trajectory $\mathbf{u}_\mathrm{H}$ and is defined as:

$$\begin{aligned} \mathbf{u}_\mathrm{R}^*(x^0, \mathbf{u}_\mathrm{H}) = \max_{\mathbf{u}_\mathrm{R}} \quad & R_\mathrm{R}(x^0, \mathbf{u}_\mathrm{R}, \mathbf{u}_\mathrm{H}), \\ \text{s.t.} \quad x^{\tau+1} = \tilde{f}(x^\tau, u_\mathrm{R}^\tau, u_\mathrm{H}^\tau), \quad & \tau \in \{0, \dots, T\}, \end{aligned} \tag{4.7}$$

where the robot's running reward is denoted by $R_{\mathrm{R}}(x^0, \mathbf{u}_{\mathrm{R}}, \mathbf{u}_{\mathrm{H}}) := \sum_{\tau=0}^{T} r_{\mathrm{R}}(x^\tau, u_{\mathrm{R}}^\tau, u_{\mathrm{H}}^\tau)$.

Finally, the parameter $\beta \in [0, \infty)$ encodes the human's rationality and governs how optimally the human behaves according to their objective; as $\beta \rightarrow 0$, the human appears "irrational", choosing their trajectory uniformly at random and ignoring any modeled structure, while $\beta \rightarrow \infty$ models the human as a perfect optimizer of the game.

4.3 Confidence-aware Role Inference for Safe Human-Robot Interaction

We propose that robots estimate the degree to which humans abide by general-sum interaction models and adapt their safety monitors accordingly. By reducing the unsafe states proportional to the observed influence between the human and robot, the robot can smoothly shift between less conservative safety monitors when the human's behavior is well-explained by the general-sum model, and the full worst-case safety monitor when the human model degrades.

Role-parameterized human model. We treat the role of the human in the general-sum game as well as their apparent rationality as hidden states. Let $\lambda \in \{\text{follow}, \text{lead}\}$ be a discrete latent variable which encodes the role of the human as either a follower or leader. To assess if the observed human behavior matches our general-sum model, we follow prior work on "human-in-isolation" models in reinterpreting the human's rationality parameter β as an indicator of model confidence [74, 34]. However, to infer both the role of the human as well as the degree to which the human is playing a general-sum game at all, we let the robot jointly infer (λ, β) given observations.

Online, after observing the current joint state x^0, the robot solves two open-loop general-sum Stackelberg games where the human swaps roles. Our final stochastic human model is:

$$P(\mathbf{u}_{\mathrm{H}} \mid x^0, \mathbf{u}_{\mathrm{R}}; \lambda, \beta) = \begin{cases} P(\mathbf{u}_{\mathrm{H}} \mid x^0, \mathbf{u}_{\mathrm{R}}; \beta), & \lambda = \text{follow} \\ P(\mathbf{u}_{\mathrm{H}} \mid x^0; \beta), & \lambda = \text{lead} \end{cases} \tag{4.8}$$

where each sub-model is computed via solving (4.6) and (4.5).

Remark: Computing the partition function in either of the two models from (4.8) requires integrating over the space of possible trajectories, Π, which is infinite. Furthermore, due to the nested optimization in the partition function when the human is a leader, approximation techniques based on Laplace approximation are not applicable. Instead, a finite choice set ($\widetilde{\Pi} \subset \Pi$) sampled from a background distribution is often exploited [70, 224]. In our experiments, we use a simple generative model based on second order polynomials to synthesize the human-driven car's likely acceleration and steering profiles at each time step to construct $\widetilde{\Pi}$.

Confidence-aware role inference. At each time step, the robot maintains a joint belief over the leader-follower roles and model confidence. Let the time horizon over which we observe trajectory snippets to be T seconds and the current time to be denoted by 0. Starting from the past state $\hat{x}$, the robot observes its executed trajectory $\mathbf{u}_\mathrm{R} := [u_\mathrm{R}^{-T}, \dots, u_\mathrm{R}^{0}]^\top$ over the time interval $[-T, 0]$ and the human's behavior $\mathbf{u}_\mathrm{H}$ over the same interval. Using these and the model (4.8), the robot updates its belief about (λ, β) via a Bayesian update:

$$b'(\lambda, \beta \mid \hat{x}, \mathbf{u}_\mathrm{H}, \mathbf{u}_\mathrm{R}) \propto P(\mathbf{u}_\mathrm{H} \mid \hat{x}, \mathbf{u}_\mathrm{R}; \lambda, \beta) b(\lambda, \beta). \tag{4.9}$$

Since we will use the belief to modify the set of unsafe states, the update should be very fast. In theory, $\beta \in [0, \infty)$ is a continuous random variable, posing computational challenges when solving (4.9). Following [74], we maintain a belief over a relatively small set of discrete β values (see Section 4.5).

Online update of the safety monitor. To modulate the conservatism of the BRT, we will look to the current belief over the human's role and the model confidence to weight the likelihood of future human control trajectories. Specifically, we marginalize over (λ, β) according to the current belief $b(\lambda, \beta)$ to obtain a distribution over human trajectories starting from the current state x^0:

$$P(\mathbf{u}_\mathrm{H} \mid x^0, \mathbf{u}_\mathrm{R}) = \mathbb{E}_{\lambda, \beta \sim b(\lambda, \beta)} \left[P(\mathbf{u}_\mathrm{H} \mid x^0, \mathbf{u}_\mathrm{R}; \lambda, \beta) \right]. \tag{4.10}$$

This model enables us to determine which future control profiles the robot can expect to see from the human in response to the robot's own motion plan. To leverage our modelled structure and reduce conservatism, we prune away human control trajectories that are sufficiently unlikely under the marginal distribution; let this set be $\mathbb{U}_\mathrm{H}(\mathbf{u}_\mathrm{R}) := \{\mathbf{u}_\mathrm{H} : P(\mathbf{u}_\mathrm{H} \mid x^0, \mathbf{u}_\mathrm{R}) > \epsilon\}$, where ϵ is a hyperparameter that controls how many unlikely trajectories get pruned.

Finally, we must transform the set of likely control trajectories into instantaneous control bounds to compute the unsafe set. For simplicity, we have the robot safeguard against the maximum and minimum controls[3] the human could execute during *any* of the likely trajectories in $\mathbb{U}_\mathrm{H}(\mathbf{u}_\mathrm{R})$. Although this approach introduces some conservatism into the estimated human control bounds (and the resulting unsafe states), it does provide an additional layer of robustness to the exact way in which people execute their local motions. Our results in Section 4.5 additionally indicate useful reduction in the size of the unsafe sets for a variety of scenarios. Let the restricted set of human controls be

$$\widetilde{\mathcal{U}}_\mathrm{H} := \left[\underline{u}_\mathrm{H}(\mathbf{u}_\mathrm{R}), \overline{u}_\mathrm{H}(\mathbf{u}_\mathrm{R}) \right],$$

where

$$\underline{u}_\mathrm{H}(\mathbf{u}_\mathrm{R}) = \min_{\tau \in \{0, \dots, T\}} \mathbb{U}_\mathrm{H}(\mathbf{u}_\mathrm{R}), \quad \overline{u}_\mathrm{H}(\mathbf{u}_\mathrm{R}) = \max_{\tau \in \{0, \dots, T\}} \mathbb{U}_\mathrm{H}(\mathbf{u}_\mathrm{R})$$

are the lower and upper control bounds respectively.

The final set of restricted human controls $\widetilde{\mathcal{U}}_\mathrm{H}$ is ultimately used in the computation of the BRT when evaluating the Hamiltonian in Eq. (4.2). By solving the HJI-VI with

[3] A more computationally costly time-varying restriction is also possible.

these restricted human control bounds, obtaining the corresponding value function, and computing the unsafe states via the sub-zero level set from (4.3), we finally obtain the restricted *Bayesian BRT*, $\widetilde{\mathcal{V}}(\tau)$. Note that this set of unsafe states is recomputed[4] after each belief update, since the belief affects the human's control bounds used in the reachability computation and influences the size and shape of $\widetilde{\mathcal{V}}(\tau)$.

Ultimately, as the robot's confidence-aware role estimate evolves, the robot automatically shifts between unsafe sets of various sizes. When the observed human motion can be well-described by the general-sum game, the marginalized set of human control profiles will place larger probability mass on those trajectories which are optimal under that model (e.g., if the human is confidently estimated to be a follower, then trajectories where the human slows down to let the robot pass). This in turn *shrinks* the estimated human control bounds used during the BRT computation since the likely maximum and minimum controls are structured according to the general-sum game and not according to what is dynamically feasible for the human to execute.

However, as the human's behavior deviates from the modelled structure, the robot's model confidence will be low for *all* human roles; as all the probability mass concentrates on low values of β, the marginalized distribution from (4.10) approaches a uniform distribution over all trajectories (irrespective of the leader-follower structure). This automatically *enlarges* the human's control bounds (and the resulting BRT) since the distribution indicates that the likely control trajectories could exhibit any dynamically-feasible behavior.

4.4 Experimental Setup

Human-robot system dynamics. In our simulation experiments, we consider a dynamical system that encodes the relative dynamics between the robot car and the human-driven car in a pairwise interaction [134, 138]. The human-driven car is modelled as an extended unicycle model and the robot car is modelled with a high-fidelity bicycle model [171]. The state of the relative system is $x_{\text{rel}} = \left[p^x_{\text{rel}}, p^y_{\text{rel}}, \psi_{\text{rel}}, v_{\text{R}}, v_{\text{H}}\right]^\top$, where $p^x_{\text{rel}}, p^y_{\text{rel}}$ are the x and y-coordinate of the human-driven car in the coordinate frame centered at the geometric center of the robot car with x-axis aligned with the heading of the self-driving car, ψ_{rel} denotes the relative heading between the two cars, and v_{R} (resp., v_{H}) denotes the speed of the robot car (resp., human-driven car). Let $u_{\text{R}} = \left[a_{\text{R}}, \delta_f\right]$ be the robot's control input where a_{R} is the acceleration and δ_f is the front wheel rotation and $u_{\text{H}} = \left[a_{\text{H}}, \omega_{\text{H}}\right]$ denote the human's control input where a_{H} is the acceleration and ω_{H} is angular speed. The evolution of the relative system is governed by the differential equation $\dot{x}_{\text{rel}} = f_{\text{rel}}\left(x_{\text{rel}}, u_{\text{R}}, u_{\text{H}}\right)$ where: $\dot{p}^x_{\text{rel}} = \frac{v_{\text{R}} p^y_{\text{rel}}}{l_r}\sin(\beta_r) + v_{\text{H}}\cos(\psi_{\text{rel}}) - v_{\text{R}}\cos(\beta_r)$, $\dot{p}^y_{\text{rel}} = -\frac{v_{\text{R}} p^x_{\text{rel}}}{l_r}\sin(\beta_r) +$

[4] In practice, we pre-computed a bank of BRTs using various control bounds and query the BRT associated with $\widetilde{\mathcal{U}}_{\text{H}}$ online.

$v_\mathrm{H}\sin(\psi_\mathrm{rel}) - v_\mathrm{R}\sin(\beta_r)$, $\dot{\psi}_\mathrm{rel} = \omega_\mathrm{H} - \frac{v_\mathrm{R}}{l_r}\sin(\beta_r)$, $\dot{v}_\mathrm{R} = a_\mathrm{R}$, and $\dot{v}_\mathrm{H} = a_\mathrm{H}$. Here, l_f (resp., l_r) denotes the front (resp., rear) axle length of the robot car and β_r is computed via $\beta_r = \tan^{-1}(\frac{l_r}{l_r+l_f}\tan(\delta_f))$.

Human reward function. We model the human's reward function as a linear combination of predefined features [162], including the human's: 1) *speed*: desire to reach the speed limit; 2) *comfort*: preference for smooth motions; 3) *reference path deviation*: tendency to follow a reference path (e.g., center lane) in structured roads; 4) *progress*: desire to reach their goal state; 5) *safety*: collision-avoidance objective. The weights on these features are obtained by inverse reinforcement learning [237] on a human driving data set [230]. More details can be found in [205].

Hyper-parameters. The horizon for solving the HJI-VI from (4.1) and the general-sum games (4.6) and (4.5) is $T = 2$[s] with a sampling period of $\Delta t = 0.2$[s]. We discretize the confidence parameter into $\beta \in \{0.1, 1, 20\}$, and set the threshold for rejecting unlikely human trajectories to $\epsilon = 0.001$. [5]

Robot planner. The robot plans via model-predictive control [40] and leverages a game-theoretic predictive model of the human behavior as in [187, 73, 194]. At each time step, the robot computes an open-loop control trajectory $\mathbf{u}_\mathrm{R}^*$ by solving: $\max_{\mathbf{u}_\mathrm{R}} \mathbb{E}_{\mathbf{u}_\mathrm{H}}\left[R_\mathrm{R}(x^0, \mathbf{u}_\mathrm{R}, \mathbf{u}_\mathrm{H}) \mid P(\mathbf{u}_\mathrm{H} \mid x^0, \mathbf{u}_\mathrm{R})\right]$ where the conditional expectation is taken over game-theoretic human trajectories following (4.5) with a fixed rationality. The robot applies the first control in the trajectory and re-plans at the next time in a receding-horizon fashion. Note that as per prior work in game-theoretic planning, the human is assumed to have a fixed follower role in the interaction at all times. While this introduces some modelling error, it nevertheless enables interesting robot behaviors and is an example of how not all motion planners will be perfect, underscoring the need for monitoring safety of the actual human-robot system.

Safety controller. Given a BRT, the robot employs a switching control strategy to avoid unsafe situations [25]; this strategy is agnostic to the upstream robot planner which may be arbitrarily complex. Specifically, when the human-driven car reaches the boundary of the BRT, the planner controls are overridden by the safety controller from (4.4).

4.5 Simulated Human-Robot Interaction Results

We first investigate two core aspects of our proposed method and the overall human-robot system: (a) the ability of our Bayesian BRT to modulate its size based on the confidence-aware game-theoretic model given a range of simulated human behavior, (b) the effect of our safety method on overall robot behavior. We additionally perform two ablation studies to assess the value of incorporating both model confidence and influence models into safety monitors.

[5]In principle, $\epsilon < \frac{1}{|\tilde{\Gamma}|}$ so the Bayesian BRT safeguards against worst-case human behaviors when the model cannot explain the data.

Round-about Merging Scenario					
	Full BRT		Bayes BRT		
Human type	CR	SOR	CR	SOR	RIP(Full)
modeled	0	23.3	0	4.7	**27.75 ± 4.03**
noisy	0	29.8	0	7.3	**18.26 ± 3.96**
unmodeled	0	42.1	0	41.7	0.06 ± 0.19

Highway Scenario					
	Full BRT		Bayes BRT		
Human type	CR	SOR	CR	SOR	RIP(Full)
modeled	0	28.3	0	9.2	**24.26 ± 6.16**
noisy	0	43.2	0	17.4	**14.83 ± 4.22**
unmodeled	0	64.8	0	62.3	0.13 ± 0.08

Table 4.1: Results for the round-about and highway; each row is averaged across 20 interactions. CR: collision rate; SOR: safety override rate; RIP: reward improvement %.

Simulated humans. We simulate three types of humans: (*modeled*) a rational Stackelberg human, (*noisy*) a suboptimal Stackelberg human, and (*unmodeled*) a non-Stackelberg human driving with constant controls (e.g., distracted driver). The Stackelberg human's role (leader or follower) is randomly assigned.

Metrics. We evaluate the safety performance, conservatism, and overall reward of the robot's motion plan when relying on various safety monitors. We measure: 1) *collision rate (CR)*: average number interactions in which the human and robot collide; 2) *safety override rate (SOR)*: average number of time steps during a finite-horizon trajectory when the safety controller is activated; 3) *reward improvement percent (RIP)*: the percent reward increase in the robot's executed trajectory when it uses our Bayesian BRT as compared to using a baseline BRT method (where the appropriate baseline depends on our case study). Mathematically, for any baseline safety method i,

$$\text{RIP}(i) := \frac{R_{\text{R}}(x^0, \mathbf{u}_{\text{R}}{}^{\text{Bayes}}, \mathbf{u}_{\text{H}}{}^{\text{Bayes}}) - R_{\text{R}}(x^0, \mathbf{u}_{\text{R}}{}^{i}, \mathbf{u}_{\text{H}}{}^{i})}{|R_{\text{R}}(x^0, \mathbf{u}_{\text{R}}{}^{i}, \mathbf{u}_{\text{H}}{}^{i})|},$$

where x^0 is the initial condition of the simulation, $\mathbf{u}_{\text{R}}{}^{\text{Bayes}}$ is the robot's executed control when using our Bayesian BRT, $\mathbf{u}_{\text{R}}{}^{i}$ is the robot's executed trajectory when using the baseline BRT, and $\mathbf{u}_{\text{H}}{}^{i}$, $\mathbf{u}_{\text{H}}{}^{\text{Bayes}}$ denote the corresponding human trajectories.

4.5.1 On the utility of the Bayesian BRT vs. full BRT

We first compare the performance of our Bayesian BRT to the full BRT (where the unsafe sets are computed with respect to all dynamically-feasible human controls) for each type of

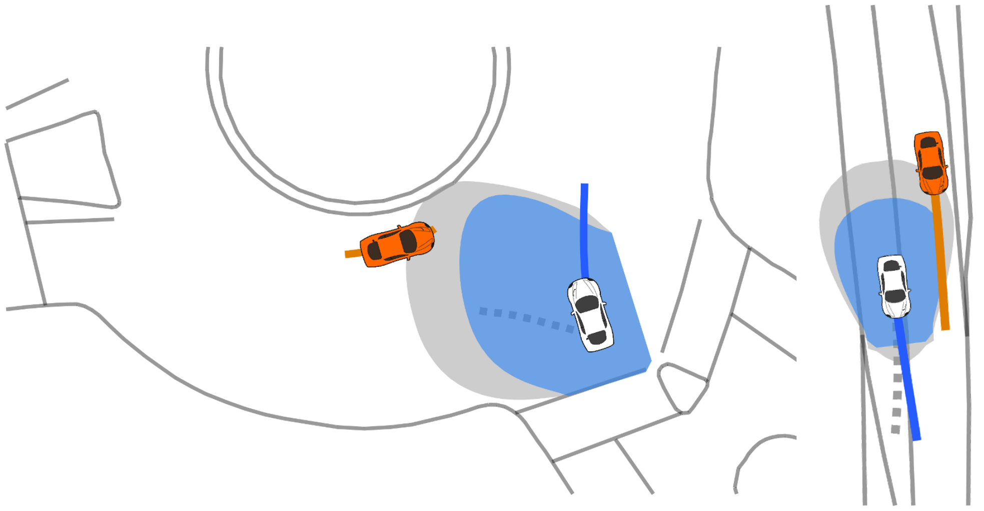


Figure 4.2: Robot car interacts with a simulated human *follower*. Our Bayes BRT infers that the human is game-theoretic and tends to cooperate; it reduces the size of the unsafe set (filled blue) accordingly. A 2-D slice of the full 5-D unsafe set is visualized in the p^x_{rel} and p^y_{rel} dimensions with all other states held fixed at the current values. Dotted gray line is the robot's trajectory when using the full BRT (filled gray). Note that the full BRT results in the robot executing unnecessary safety maneuver in (a) and aborting merging in (b).

simulated human. We compute our metrics in two traffic scenarios, round-about merging and highway lane-change, across 20 interactions for each test. Our findings for both scenarios are summarized in Tab. 4.1 and snapshots of the full BRT and our Bayesian BRT are visualized in Fig. 4.2.

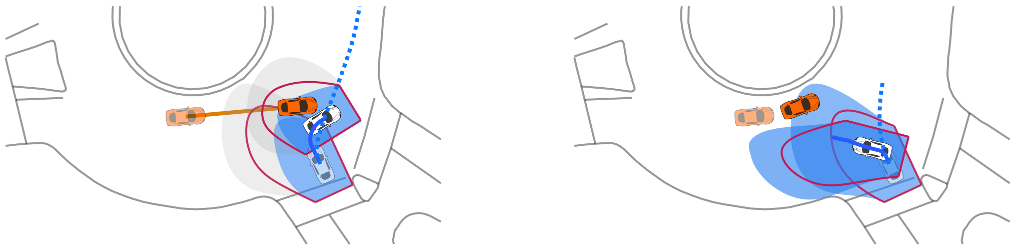
Results from both scenarios indicate that when the true simulated human behaves according to the general-sum Stackelberg game, our Bayesian BRT can detect this modelled structure and reduce the conservatism of the safety guarantee (e.g., in the round-about, resulting in ∼20% less safety controller activations than the full BRT) while preserving collision-free human-robot interaction. Furthermore, the overall reward of the robot's executed trajectory is increased by 27.75% when compared to the trajectory the robot would execute if it was relying on the overly-conservative full BRT as its safety monitor. Importantly, as the simulated human behavior becomes increasingly misspecified, we also see the Bayesian BRT increasing in conservatism—ultimately approaching comparable performance across all metrics to the full BRT when the distracted human driver (*unmodeled*) behaves in a completely non-game-theoretic way.

4.5.2 Ablation Study 1: On the value of model confidence

To investigate the effect of model confidence in our method, we implement a version of our Bayesian BRT without model confidence but with the general-sum model; we call this method *λ-only Bayesian BRT*. We stress-test this model's utility in the round-about scenario when the robot interacts with a non-game-theoretic distracted human (*unmodeled*). As Tab. 4.2 indicates, trusting the model completely leads to a high collision percentage; this is because the estimated control bounds do not adequately capture the true human controls (see visualization in left of Fig. 4.3). Additionally, analyzing the reward improvement percent that our Bayesian BRT achieves when compared to the λ-only Bayesian BRT, we see a 9.18% improvement since the robot detects misspecification early and takes evasive maneuvers.

	λ-only Bayesian BRT		
Human type	CR	SOR	RIP(λ-only)
unmodeled	25	11.3	**9.18±2.9**

Table 4.2: Results without β but with game-theoretic model.



(a) (b)

Figure 4.3: (a) Failure to detect model mismatch results in collision. The λ-only BRT (filled blue) is smaller than the least-conservative BRT (pink outline, defined in Section 4.6), meaning safety is no longer guaranteed. Dotted blue line is robot trajectory using our Bayes BRT, avoiding collision. (b) Failure to use an interactive human model results in the robot avoiding the human even when they try to cooperate.

4.5.3 Ablation Study 2: On the value of game-theoretic models

To understand the utility of general-sum game-theoretic models in our safety monitor, we implement a version of our Bayesian BRT with a human-in-isolation model but with model confidence (as in [74]): we call this method *β-no game Bayesian BRT*. We stress-test this method against a simulated human which behaves rationally according to the game-theoretic interaction model. Results are shown in Tab. 4.3. Because the underlying human model in *β-no game Bayesian BRT* assumes that people are not influenced by the robot, then the model confidence is always low in highly dynamic interactions with the human (see right of Fig. 4.3). This results in the safety monitor relying on the full set of human controls and, although the robot never collides, it activates its safety monitor near identically as often as the full BRT interacting with a *modeled* human from Tab. 4.1. Our Bayesian BRT improves the robot's reward by ~29% over the *β-no game Bayesian BRT* since the robot can confidently execute its plan without the safety monitor intervening.

	β-no game Bayes BRT		
Human type	CR	SOR	RIP(β-no game)
modeled	0	24.7	**29.18 ± 3.63**

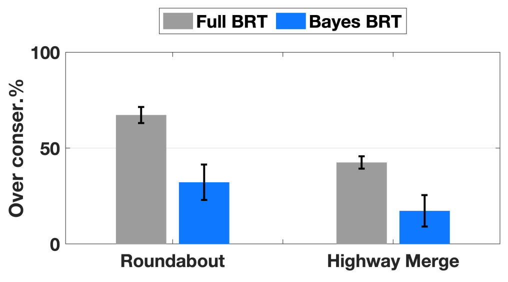
Table 4.3: Results with β but no game-theoretic model.

4.6 Evaluation with Real Traffic Data

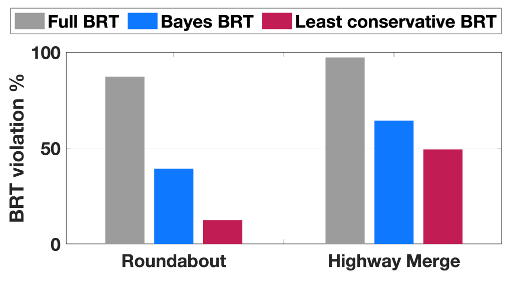
We investigate how the full BRT and our Bayesian BRT perform *ex post facto* on recorded human traffic data. We extract 200 pair-wise interactions from [230] and for each interaction we assign one car as the robot car (the other as the human), and run our approach of constructing the Bayesian BRT while replaying the two cars' recorded trajectories. We compare our method to two baselines: (1) the full BRT and (2) the least-conservative BRT, $\mathcal{V}^*(\tau)$, which is obtained by restricting the human's control set in the HJI-VI to be the maximum and minimum controls observed in a T-length snippet of the ground-truth human trajectory starting at the current timestep. Theoretically, this is the least conservative unsafe set that can be obtained via our approach.

At each timestep, we measure the *over-conservatism percentage* of a given BRT $\mathcal{V}^i(\tau)$, $i \in \{\text{Full}, \text{Bayes}\}$, by comparing the area of $\mathcal{V}^i(\tau)$ to the least-conservative BRT $\mathcal{V}^*(\tau)$. Mathematically, this is $\frac{\mathcal{A}(\mathcal{V}^i(\tau))-\mathcal{A}(\mathcal{V}^*(\tau))}{\mathcal{A}(\mathcal{V}^*(\tau))}$, where the function $\mathcal{A}$ maps a set of states to its geometric area. We also measure the *BRT violation percentage*: the percentage of interactions flagged as unsafe by a given BRT method. Note that once the BRT is breached, trajectory replay stops.

Our findings are summarized in 4.4, showing the average over-conservatism percentage and average BRT violation percentage for each traffic scenario. Both the over-conservatism and the BRT violation percentage of our Bayesian BRT are significantly reduced compared to the full BRT. This suggests that human driving is closer, from a safety perspective, to our method than to the traditional full BRT approach. However, humans are still violating the constraints our method would impose, meaning there is still room to improve efficiency while preserving the same safety levels as humans. Finally, note that the Bayesian BRT's over-conservatism percentage is positive, indicating that our method is more conservative than the least-conservative BRT, preserving the quality of our safety monitor.



(a)



(b)

Figure 4.4: (left) Our method is not overly-conservative compared to the least-restrictive BRT. (right) Our method decreases safety violations vs. the full BRT.

4.7 Discussion & Conclusion

We proposed that robot safety monitors be imbued with confidence-aware game-theoretic models. By restricting the set of feasible human controls based on how much the human follows the game-theoretic model, the robot can automatically interpolate between smaller unsafe sets consistent with the human model and the full worst-case unsafe set.

Our traffic data experiments revealed that even the least-conservative BRT is more conservative than real drivers; this is due to our control bound construction and the zero-sum nature of the reachability game. We are excited for future work on new safety methods which reflect human notions of safety. Further, as in [134], designing robot planners which are aware of the safety monitor is an interesting future direction.

Chapter 5

Scalable Multi-Human, Multi-Robot Collision Avoidance

This chapter is based on the paper "A scalable framework for real-time multi-robot, multi-human collision avoidance" [15] written in collaboration with Sylvia Herbert, David Fridovich-Keil, Jaime Fisac, Sampada Deglurkar, Anca Dragan, and Claire Tomlin.

As robotic systems are increasingly used for applications such as drone delivery services, semi-automated warehouses, and autonomous cars, safe and efficient robotic navigation around multiple humans is crucial. Consider the example in Fig. 5.1, inspired by a drone delivery scenario, where two quadcopters must plan a safe trajectory around two humans who are walking through the environment. We would like to guarantee that the robots will reach their goals without ever colliding with each other, any of the humans, or the static surroundings.[1]

This safe motion planning problem faces three main challenges: (1) controlling the nonlinear robot dynamics subject to external disturbances (e.g. wind), (2) planning around multiple humans in real time, and (3) avoiding conflicts with other robots' plans. Extensive prior work from control theory, motion planning, and cognitive science has enabled computational tools for rigorous safety analysis, faster motion planners for nonlinear systems, and predictive models of human agents. Individually, these problems are difficult—computing robust control policies, coupled robot plans, and joint predictions of multiple human agents are all computationally demanding at best and intractable at worst [158, 49]. Recent work, however, has made progress in provably-safe real-time motion planning [95, 154, 200], real-time probabilistic prediction of a human agent's motion [74, 238], and robust sequential trajectory planning for multi-robot systems [26, 48]. It remains a challenge to synthesize these into a real-time planning system, primarily due to the difficulty of joint planning and prediction for multiple robots and humans. There has been some work

[1]Note that our laboratory setting uses a motion capture system for sensing and state estimation—robustness with respect to sensor uncertainty is an important component that is beyond the scope of this paper.

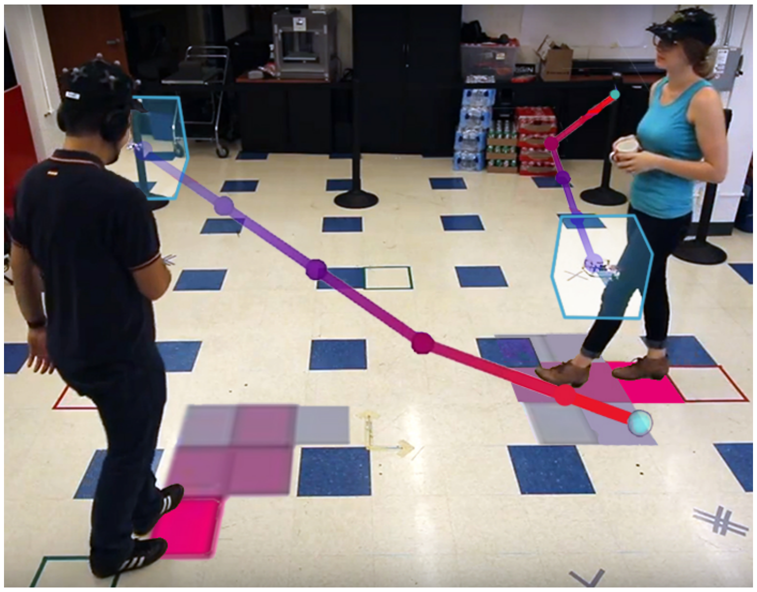


Figure 5.1: Hardware demonstration of real-time multi-agent planning while maintaining safety with respect to internal dynamics, external disturbances, and intentional humans. The planned trajectories from the quadcopters are visualized, and the tracking error bound is shown as a box around each quadcopter. The probabilistic distribution over the future motion of the humans are shown in pink in front of each human.

combining subsets of this problem [118, 214, 126], but the full setup of real-time and robust multi-robot navigation around multiple humans remains underexplored.

Our main contributions in this paper are tractable approaches to joint planning and prediction, while still ensuring efficient, probabilistically-safe motion planning. We use the reachability-based FaSTrack framework [95] for real-time robust motion planning. To ensure real-time feasibility, robots predict human motion using a simple model neglecting future interaction effects. Because this model will be a simplification of true human motion, we use confidence-aware predictions [74] that become more conservative whenever humans deviate from the assumed model. Finally, groups of robots plan sequentially according to a pre-specified priority ordering [51], which serves to reduce the complexity of the joint planning problem while maintaining safety with respect to each other. We demonstrate our framework in hardware, and provide a large-scale simulation to showcase scalability.

5.1 The SCAFFOLD Framework

Fig. 5.2 illustrates our overall planning framework, called SCAFFOLD. We introduce the components of the framework by incrementally addressing the three main challenges identified above.

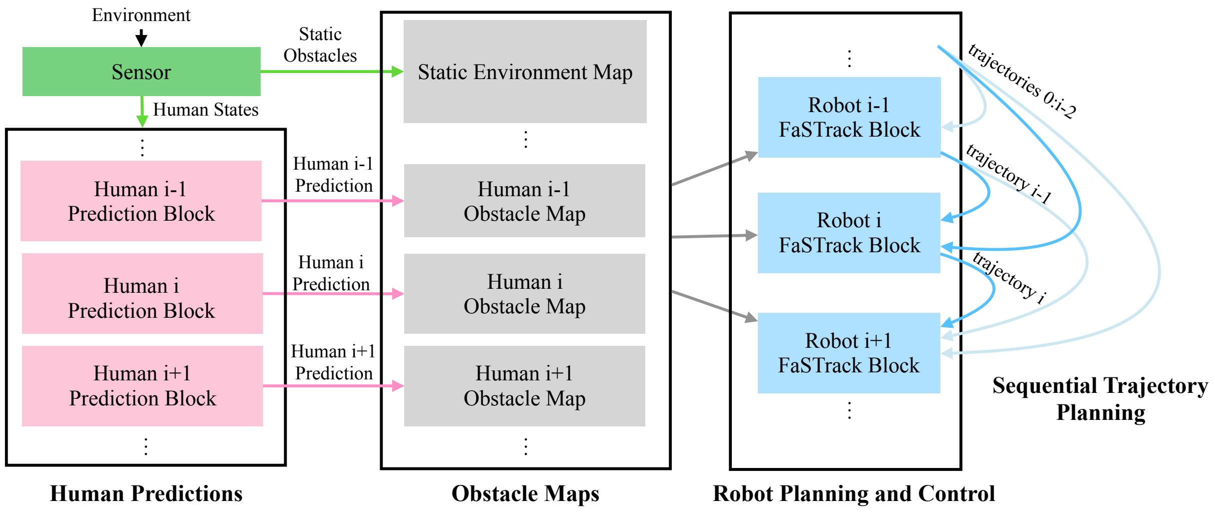


Figure 5.2: The SCAFFOLD Framework

We first present the **robot planning and control** block (Section 5.2), which is instantiated for each robot. Each robot uses a robust controller (e.g. the reachability-based controller of [95]) to track motion plans within a precomputed error margin that accounts for modeled dynamics and external disturbances. In order to generate safe motion plans, each robot will ensure that output trajectories are collision-checked with a set of obstacle maps, using the tracking error margin.

These obstacle maps include an *a priori* known set of static obstacles, as well as predictions of the future motion of any humans, which are generated by the **human predictions** block (Section 5.3). By generating these predictions, each robot is able to remain probabilistically safe with respect to the humans. To ensure tractability for multiple humans, we generate predictions using simplified interaction models, and subsequently adapt them following a real-time Bayesian approach such as [74]. We leverage the property that individual predictions automatically become more uncertain whenever their accuracy degrades, and use this to enable our tractable predictions to be robust to unmodeled interaction effects.

Finally, to guarantee safety with respect to other robots, we carry out **sequential trajectory planning** (Section 5.4) by adapting the cooperative multi-agent planning scheme [26] to function in real time with the robust trajectories from the planning and control block. The robots generate plans according to a pre-specified priority ordering. Each robot plans to avoid the most recently generated trajectories from robots of higher priority, i.e. robot i must generate a plan that is safe with respect to the planned trajectories from robots $j, j < i$. This removes the computational complexity of planning in the joint state space of all robots at once.

5.2 Robot Planning and Control

In this section we begin with the canonical problem of planning through a static environment. Efficient algorithms such as A* or rapidly-exploring random trees (RRT) [92, 110] excel at this task. These algorithms readily extend to environments with deterministically-moving obstacles by collision-checking in both time and space.

We now introduce robot dynamics and allow the environment to have external disturbances such as wind. Kinematic planners such as A* and RRT do not consider these factors when creating plans. In practice, however, these planners are often used to generate an initial trajectory, which may then be smoothed and tracked using a feedback controller such as a linear quadratic regulator (LQR). During execution, the mismatch between the planning model and the physical system can result in tracking error, which may cause the robot to deviate far enough from its plan to collide with an obstacle. To reduce the chance of collision, one can augment the robot by a heuristic error buffer; this induces a "safety bubble" around the robot used when collision checking. However, heuristically generating this buffer will not guarantee safety.

Several recent approaches address efficient planning while considering model dynamics and maintaining robustness with respect to external disturbances. Majumdar and Tedrake [154] use motion primitives with safety funnels, while Raković [172] utilizes robust model-predictive control, and Singh et. al. [199] leverage contraction theory.

In this paper, we use FaSTrack [95, 78], a modular framework that computes a tracking error bound (TEB) via *offline* reachability analysis. This TEB can be thought of as a rigorous counterpart of the error-buffer concept introduced above. More concretely, the TEB is the set of states capturing the maximum relative distance (i.e. maximum tracking error) that may occur between the physical robot and the current state of the planned trajectory. We compute the TEB by formulating the tracking task as a pursuit-evasion game between the planning algorithm and the physical robot. We then solve this differential game using Hamilton-Jacobi reachability analysis. To ensure robustness, we assume (a) worst-case behavior of the planning algorithm (i.e. being as difficult as possible to track), and (b) that the robot is experiencing worst-case, bounded external disturbances. The computation of the TEB also provides a corresponding error-feedback controller for the robot to always remain inside the TEB. Thus, FaSTrack wraps efficient motion planners, and adds robustness to modeled system dynamics and external disturbances through the precomputed TEB and error-feedback controller. Fig. 5.4 shows a top-down view of a quadcopter using a kinematic planner (A*) to navigate around static obstacles. By employing the error-feedback controller, the quadcopter is guaranteed to remain within the TEB (shown in blue) as it traverses the A* path.

5.2.1 FaSTrack Block

Requirements: To use FaSTrack, one needs a high-fidelity dynamical model of the system used for reference tracking, and a (potentially simpler) dynamic or kinematic

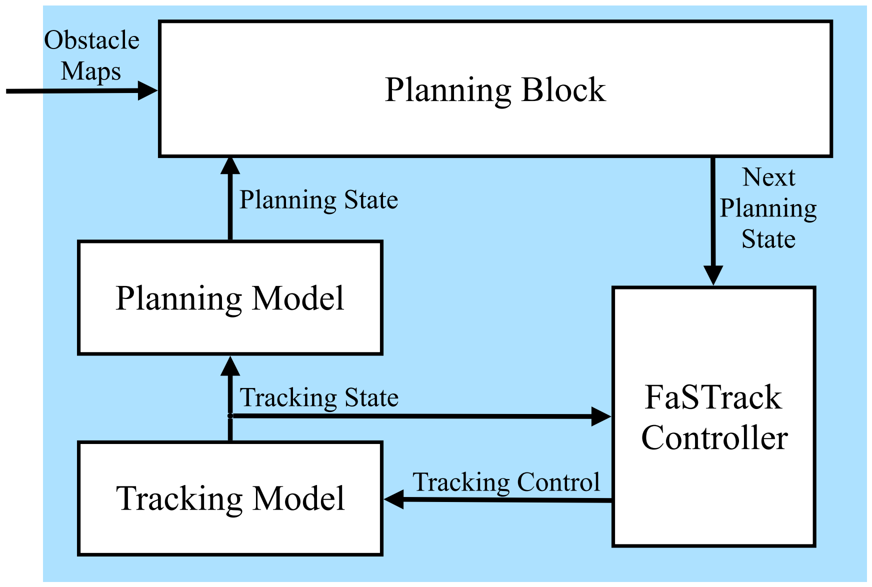


Figure 5.3: FaSTrack Block

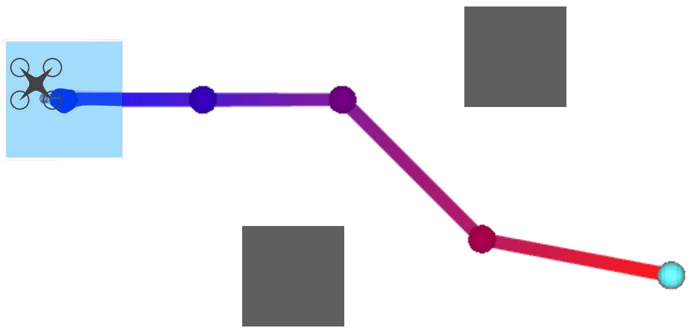


Figure 5.4: Top-down view of FaSTrack applied to a 6D quadcopter navigating a static environment. Note the simple planned trajectory (changing color over time) and the tracking error bound (TEB) around the quadcopter. This TEB is a 6D set that has been projected down to the position dimensions. Because we assuem the quadcopter moves independently in (x, y, z), this projection looks like a box, making collision-checking very straightforward.

model used by the planning algorithm. Using the relative dynamics between the tracking model and the planning model, the TEB and safety controller may be computed using Hamilton-Jacobi reachability analysis [95], sum-of-squares optimization [200], or approximate dynamic programming [179].

Implementation: Fig. 5.3 describes the online algorithm for FaSTrack after the offline precomputation of the TEB and safety controller. We initialize the **planning block** to start within the TEB centered on the robot's current state. The planner then uses any desired planning algorithm (e.g. A*, or model predictive control) to find a trajectory from this initial state to a desired goal state. When collision-checking, the planning algorithm must ensure that the tube defined by the Minkowski sum of the TEB and the planned trajectory does not overlap any obstacles in the **obstacle map**.

The planning block provides the current planned reference state to the **FaSTrack controller**, which determines the relative state between the tracking model (robot) and planned reference (motion plan). The controller then applies the corresponding optimal, safe tracking control via an efficient look-up table.

5.2.2 FaSTrack in the SCAFFOLD Framework

In the **robot planning and control** section of Fig. 5.2, each robot uses FaSTrack for robust planning and control. FaSTrack guarantees that each robot remains within its TEB-augmented trajectory.

5.3 Human Predictions

Section 5.2 introduced methods for the fast and safe navigation of a single robot in an environment with deterministic, moving obstacles. However, moving obstacles—especially human beings—are not always best modeled as deterministic. For such "obstacles," robots can employ probabilistic predictive models to produce a distribution of states the human may occupy in the future. The quality of these predictions and the methods used to plan around them determine the overall safety of the system. Generating accurate real-time predictions for multiple humans (and, more generally, uncertain agents) is an open problem. Part of the challenge arises from the combinatorial explosion of interaction effects as the number of agents increases. Any simplifying assumptions, such as neglecting interaction effects, will inevitably cause predictions to become inaccurate. Such inaccuracies could threaten the safety of plans that rely on these predictions.

Our goal is to compute real-time motion plans that are based on up-to-date predictions of all humans in the environment, and at the same time maintain safety when these predictions become inaccurate. The confidence-aware prediction approach of [74] provides a convenient mechanism for adapting prediction uncertainty online to reflect the degree to which humans' actions match an internal model. This automatic uncertainty adjustment allows us to simplify or even neglect interaction effects between humans, because uncertain predictions will automatically result in more conservative plans when the observed behavior departs from internal modeling assumptions.

5.3.1 Human Prediction Block

Requirements: In order to make any sort of collision-avoidance guarantees, we require a prediction algorithm that produces distributions over future states, and rapidly adjusts those predictions such that the actual trajectories followed by humans lie within the prediction envelope. There are many approaches to probabilistic trajectory prediction in the literature, e.g. [62, 94, 238, 133]. These methods could be used to produce a probabilistic prediction of the i-th human's state $x_i \in \mathbb{R}^{n_i}$ at future times τ, conditioned on observations[2] z: $P(x_i^\tau \mid z^{0:t})$. These observations are random variables and depend upon the full state of all robots and humans x until the current time t. However, by default these distributions will not necessarily capture the true trajectories of each human, especially when the models do not explicitly account for interaction effects. Fisac et. al. [74] provide an efficient mechanism for updating the uncertainty (e.g., the variance) of distributions $P(x_i^\tau | z^{0:t})$ to satisfy this safety requirement.

Implementation: Fig. 5.5 illustrates the **human prediction block**. We use a maximum-entropy model as in [74, 70, 237], in which the dynamics of the i-th human are affected by actions u_i^t drawn from a Boltzmann probability distribution. This time-dependent distribution over actions implies a distribution over future states. Given a sensed state x_i^t of human i at time t, we invert the dynamics model to infer the human's action, u_i^t. Given this action, we perform a Bayesian update on the distribution of two parameters: θ_i, which describes the objective of the human (e.g. the set of candidate goal locations), and β_i, which governs the variance of the predicted action distributions. β_i can be interpreted as a natural indicator of *model confidence*, quantifying the model's ability to capture humans' current behavior [74]. Were we to model actions with a different distribution, e.g. a Gaussian process, the corresponding parameters could be learned from prior data [237, 238, 70], or inferred online [186, 74] using standard inverse optimal control (inverse reinforcement learning) techniques.

Once distributional parameters are updated, we produce a prediction over the future actions of human i through the following Boltzmann distribution:

$$P(u_i^t \mid x^t; \beta_i, \theta_i) \propto e^{\beta_i Q_i(x^t, u_i^t; \theta_i)} \quad . \tag{5.1}$$

This model treats each human as more likely to choose actions with high expected utility as measured by the (state-action) Q-value associated to a certain reward function, $r_i(x, u_i; \theta_i)$. In general, this value function may depend upon the joint state x and the human's own action u_i, as well as the parameters θ_i, β_i. Finally, combining (5.1) with a dynamics model, these predicted actions may be used to generate a distribution over future states. In practice, we represent this distribution as a discrete occupancy grid. One such grid is visualized in Fig. 5.6.

By reasoning about each human's model confidence as a hidden state [74], our framework dynamically adapts predictions to the evolving accuracy of the models encoded in the set of state-action functions, $\{Q_i\}$. Uncertain predictions will force the planner to be more cautious whenever the actions of the humans occur with low probability as measured by (5.1).

[2]For simplicity, we will later assume complete state observability: $z^t = x^t$.

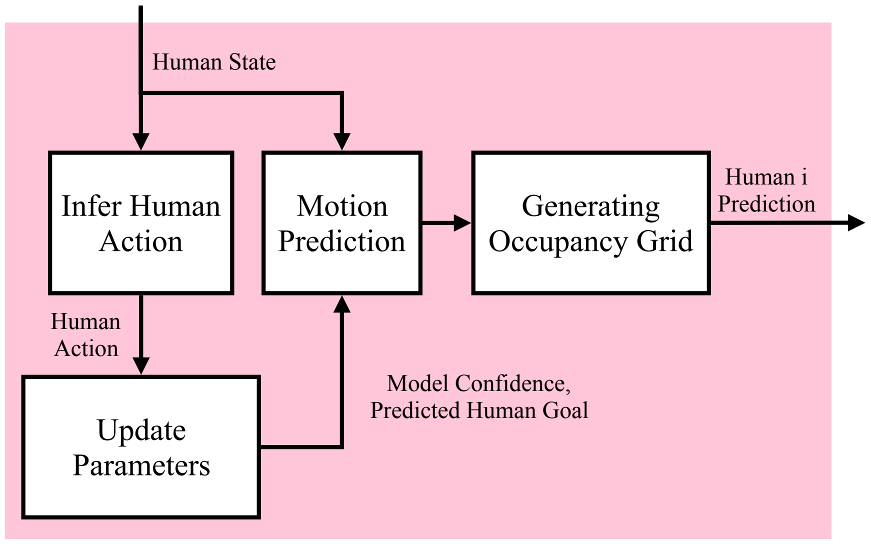


Figure 5.5: Human Prediction Block

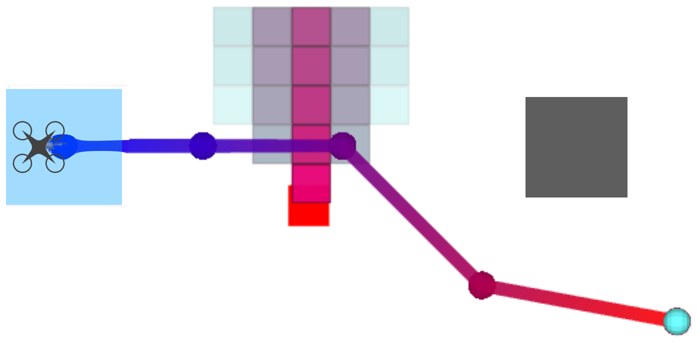


Figure 5.6: Our environment now has a human (red square). The robot models the human as likely to move north. Visualized on top of the human is the distribution of future states (pink is high, blue is low probability). Since the human is walking north and matching the model, the robot's predictions are confident that the human will continue northward and remain collision-free.

5.3.2 Human Prediction in the SCAFFOLD Framework

The predicted future motion of the humans is generated as a probability mass function, represented as time-indexed set of occupancy grids. These distributions are interpreted as an obstacle map by the FaSTrack block. During planning, a state is considered to be unsafe if the total probability mass contained within the TEB centered at that state exceeds a preset threshold, P_{th}. As in [74], we consider a trajectory to be unsafe if the maximum *marginal* collision probability at any individual state along it exceeds P_{th}.

When there are multiple humans, their state at any future time τ will generally be characterized by a joint probability distribution $P(x_1^\tau, ..., x_N^\tau)$.

Let x_R^τ be the planned state of a robot at time τ. We write $\text{coll}(x_R^\tau, x_i^\tau)$ to denote the overlap of the TEB centered at x_R^τ with the ith human at state x_i^τ. Thus, we may formalize the probability of collision with *at least one* human as:

$$\begin{aligned} &P\big(\text{coll}(x_R^\tau, \{x_i^\tau\}_{i=1}^N)\big) = \\ &\qquad 1 - \prod_{i=1}^{N} P\big(\neg\text{coll}(x_R^\tau, x_i^\tau) \mid \neg\text{coll}(x_R^\tau, \{x_j^\tau\}_{j=1}^{i-1})\big) \ , \end{aligned} \tag{5.2}$$

Intuitively, (5.2) states that the probability that the robot is in collision at s^τ is one minus the probability that the robot is not in collision. We compute the second term by taking the product over the probability that the robot is not in collision with each human, given that the robot is not in collision with all previously accounted for humans. Unfortunately, it is generally intractable to compute the terms in the product in (5.2). Fortunately, tractable approximations can be computed by storing only the marginal predicted distribution of each human at every future time step τ, and assuming independence between humans. This way, each robot need only operate with N occupancy grids. The resulting computation is:

$$P\big(\text{coll}(x_R^\tau, \{x_i^\tau\}_{i=1}^N)\big) \approx 1 - \prod_{i=1}^{N} \Big(1 - P\big(\text{coll}(x_R^\tau, x_i^\tau)\big)\Big) \ . \tag{5.3}$$

Here we take the product over the probability that the robot is not in collision with each human (one minus probability of collision), and then again take the complement to compute the probability of collision with any human. Note that when the predictive model neglects future interactions between multiple humans, (5.2) reduces to (5.3). If model confidence analysis [74] is used in conjunction with such models, we hypothesize that each marginal distribution will naturally become more uncertain when interaction effects are significant.

Once a collision probability is exactly or approximately computed, the planner can reject plans for which, at any time $\tau > t$, the probability of collision from (5.3) exceeds P_{th}. Thus, we are able to generate computationally tractable predictions that result in P_{th}-safe planned trajectories for the physical robot.

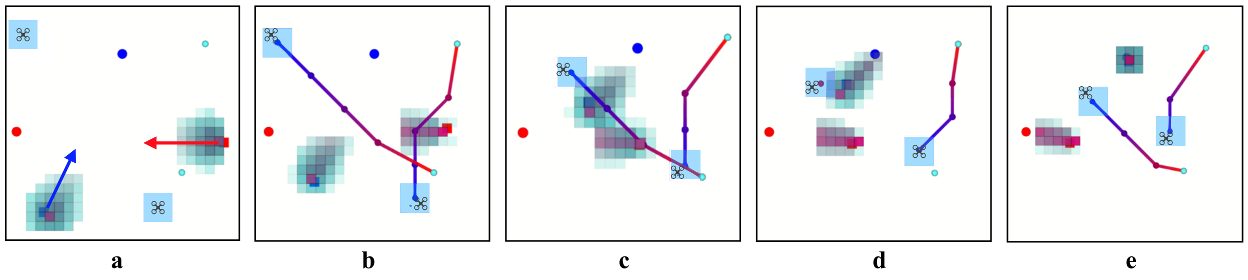


Figure 5.7: Birds-eye Robot Operating System (ROS) visualization of hardware demonstration from Fig. 9.1. (a) Two humans (red and blue) start moving towards their respective goals (also red and blue). Robot in lower right-hand corner has first priority, and robot in upper left-hand corner has second. The time-varying predictions of each human's future motion are visualized. (b) Robots plan trajectories to their goals based on the predictions, priority order, and are guaranteed to stay within the tracking error bound (shown in blue). (c) When the humans begin to interact in an unmodeled way by moving around each other, the future predictions become more uncertain. (d) The robots adjust their plans to be more conservative–note the upper-left robot waiting as the blue human moves past. (c) When the humans pass each other and the uncertainty decreases, the robots complete their trajectories.

.

5.4 Sequential Trajectory Planning

Thus far, we have shown how our framework allows a single robot to navigate in real-time through an environment with multiple humans while maintaining safety (at a probability of approximately P_{th}-safe) and accounting for internal dynamics, external disturbances, and humans. However, in many applications (such as autonomous driving), the environment may also be occupied by other robots.

Finding the optimal set of trajectories for all robots in the environment would require solving the planning problem over the joint state space of all robots. This very quickly becomes computationally intractable with increasing numbers of robots. Approaches for multi-robot trajectory planning often assume that the other vehicles operate with specific control strategies such as those involving induced velocity obstacles [223, 71, 45, 215] and involving virtual structures or potential fields to maintain collision [166, 56, 234]. These assumptions greatly reduce the dimensionality of the problem, but may not hold in general.

Rather than assuming specific control strategies of other robots, each robot could generate predictions over the future motion of all other robots. Successful results of this type typically assume that the vehicles operate with very simple dynamics, such as single integrator dynamics [235], differentially flat systems [141], linear systems [5].

However, when robots can communicate with each other, methods for centralized and/or cooperative multi-agent planning allow for techniques for scalability [137, 213, 161]. One such method is sequential trajectory planning (STP) [51], which coordinates robust multi-agent planning using a sequential priority ordering. Priority ordering is commonly used in many multi-agent scenarios, particularly for aerospace applications.

In this work, we merge STP with FaSTrack to produce real-time planning for multi-agent systems.

5.4.1 Sequential Trajectory Planning

Requirements: To apply STP, robots must be able to communicate trajectories and TEBs over a network.

Implementation: STP addresses the computational complexity of coupled motion planning by assigning a priority order to the robots and allowing higher-priority robots to ignore the planned trajectories of lower-priority robots. The first-priority robot uses the **FaSTrack block** to plan a (time-dependent) trajectory through the environment while avoiding the **obstacle maps**. This trajectory is shared across the network with all lower-priority robots. The i-th robot augments the trajectories from robots $0 : i - 1$ by their respective TEBs, and treats them as time-varying obstacles. The i-th robot determines a safe trajectory that avoids these time-varying tubes as well as the predicted state distributions of humans, and publishes this trajectory for robots $i + 1 : n$. This process continues until all robots have computed their trajectory. Each robot replans as quickly as it is able; in our experiments, this was between 50–300 ms.

5.4.2 Sequential Trajectory Planning in the SCAFFOLD Framework

By combining this method with FaSTrack for fast individual planning, the sequential nature of STP does not significantly affect overall planning time. In our implementation all computations are done on a centralized computer using the Robot Operating System (ROS), however this method can easily be performed in a decentralized manner. Note that STP does depend upon reliable, low-latency communication between the robots. If there are communication delays, techniques such as [60] may be used to augment each robot's TEB by a term relating to time delay.

5.5 Implementation and Experimental Results

We demonstrate SCAFFOLD's feasibility in hardware with two robots and two humans, and its scalability in simulation with five robots and ten humans.

5.5.1 Hardware Demonstration

We implemented the SCAFFOLD framework in C++ and Python, using Robot Operating System (ROS) [169]. All computations for our hardware demonstration were done on a laptop computer (specs: 31.3 GB of memory, 216.4 GB disk, Intel Core i7 @ 2.70GHz x 8). As shown in Fig. 5.1, we used Crazyflie 2.0 quadcopters as our robots, and two human volunteers. The position and orientation of robots and humans were measured at roughly

235 Hz by an OptiTrack infrared motion capture system. The humans were instructed to move towards different places in the lab, while the quadcopters planned collision-free trajectories in three dimensions (x, y, z) using a time-varying implementation of A*. The quadcopters tracked these trajectories using the precomputed FaSTrack controller designed for a 6D near-hover quadcopter model tracking a 3D point [78]. Human motion was predicted 2 s into the future. Fig. 5.7 shows several snapshots of this scene over time. Note that the humans must move around each other to reach their goals—this is an unmodeled interaction affect. The predictions become less certain during this interaction, and the quadcopters plan more conservatively, giving the humans a wider berth. The full video of the hardware demonstration can be viewed in our video submission.

5.5.2 SCAFFOLD Framework Simulation Analysis

Due to the relatively small size of our motion capture arena, we demonstrate scalability of the SCAFFOLD framework through a large-scale simulation. In this simulation, pedestrians are crossing through a $25 \times 20\text{m}^2$ region of the UC Berkeley campus. We simulate the pedestrians' motion using potential fields [84], which "pull" each pedestrian toward his or her own goal and "push" them away from other pedestrians and robots. These interaction effects between humans and robots are not incorporated into the state-action functions $\{Q_i\}$, and lead to increased model uncertainty (i.e., higher estimates of β_i) during such interactions. The fleet of robots must reach their respective goals while maintaining safety with respect to their internal dynamics, humans, and each other. We ran our simulation on a desktop workstation with an Intel i7 Processor and 12 CPUs operating at 3.3 GHz.[3] Our simulation took 98 seconds for all robots to reach their respective goals. Predictions over human motion took 0.15 ± 0.06 seconds to compute for each human. This computation can be done in parallel. Each robot took 0.23 ± 0.16 seconds to determine a plan. There was no significant difference in planning time between robots of varying priority. Robots used time-varying A* on a 2-dimensional grid with 1.5 m resolution, and collision checks were performed at 0.1 m along each trajectory segment. The resolution for human predictions was 0.25 m and human motion was predicted 2 s into the future.

5.6 Discussion & Conclusion

In this paper, we compose several techniques for robust and efficient planning together in a framework designed for fast multi-robot planning in environments with uncertain moving obstacles, such as humans. Each robot generates real-time motion plans while maintaining safety with respect to external disturbances and modeled dynamics via the FaSTrack framework. To maintain safety with respect to humans, robots sense humans'

[3]The computation appears to be dominated by the prediction step, which we have not yet invested effort in optimizing.

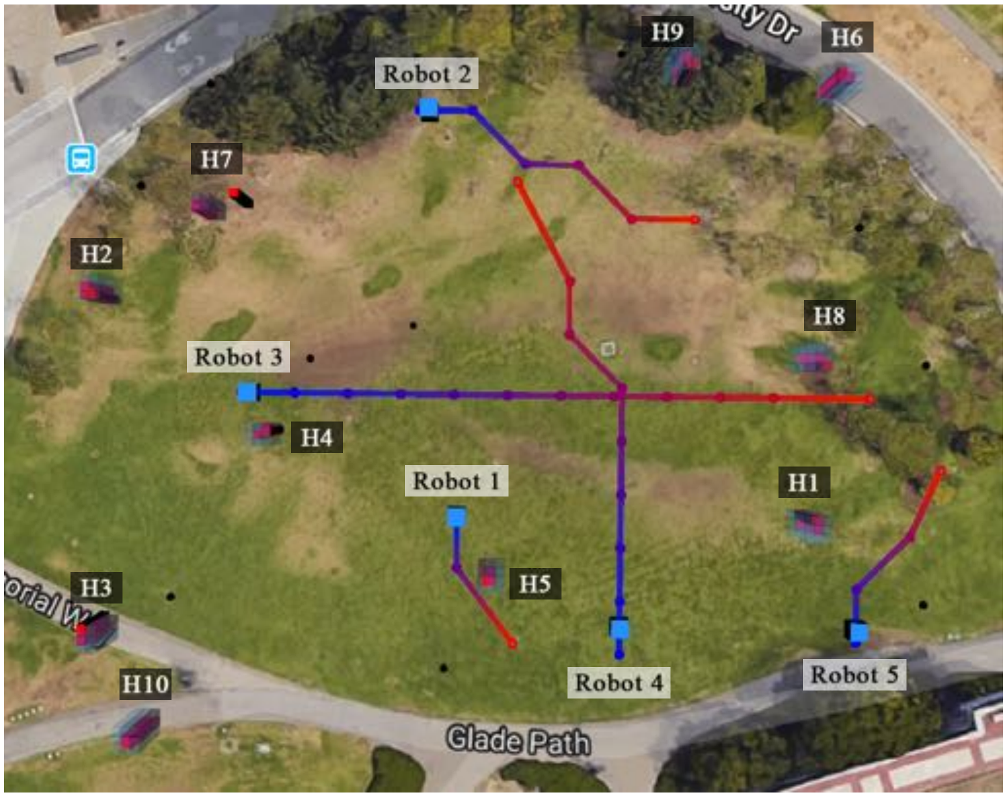


Figure 5.8: Simulation of 5 dynamic robots navigating in a scene with 10 humans. The simulated humans according to a potential field, which results in unmodeled interaction effects. However, SCAFFOLD enables each robot to still reach its goal safely.

states and form probabilistic, adaptive predictions over their future trajectories. For efficiency, we model these humans' motions as independent, and to maintain robustness, we adapt prediction model confidence online. Finally, to remain safe with respect to other robots, we introduce multi-robot cooperation through STP, which relieves the computational complexity of planning in the joint state space of all robots by instead allowing robots to plan sequentially according to a fixed priority ordering.

We demonstrate our framework in hardware with two quadcopters navigating around two humans. We also present a larger simulation of five quadcopters and ten humans.

To further demonstrate our framework's robustness, we are interested in exploring (a) non-grid based methods of planning and prediction, (b) the incorporation of sensor uncertainty, (c) optimization for timing and communication delays, and (d) recursive feasibility in planning. We are also interested in testing more sophisticated predictive models for humans, and other low-level motion planners.

Chapter 6

Robust Human Motion Prediction

This chapter is based on the paper "A robust control framework for human motion prediction" [14] written in collaboration with Somil Bansal, Ellis Ratner, Claire Tomlin, and Anca Dragan.

Robots such as autonomous vehicles and assistive manipulators are increasingly operating in dynamic environments and close physical proximity to people. In such scenarios, it is important that robots not only account for the current state of the humans nearby, but also predict their future state to plan safe and efficient trajectories.

To maximally preserve safety, a robust optimal control perspective models the human as taking any action (with equal likelihood) from a set of controls. The predictor combines this control set with a conservative human dynamics model to compute a *full forward reachable set*, or the set of all states that the human could reach from their current state [158, 67]. This approach allows the robot to produce safe predictions when very little is understood about human decision-making.

A complementary perspective is that there is structure to human decision-making: humans have intentions, and make decisions in pursuit of these intentions. For example, consider an indoor home environment where people often move towards chairs, tables, or doorways. Predictors synthesized from this perspective, called *intent-driven predictors*, build data-driven models of human actions given intent [13, 174, 238, 117, 42], and have been successful in a variety of domains including manipulation [8, 62, 123], autonomous driving [191], and navigation [151, 178] (see [183] for a survey). Since human behavior varies between people and over time, these decision-making models are often parameterized and predictors maintain a belief distribution over the model parameters [72, 132]. This provides a direct and succinct way for the robot to use online data to update its human model [238, 23, 120, 13].

However, a key challenge remains with such intent-driven predictors. To update the belief over model parameters and to generate predictions, intent-driven predictors rely on priors and on likelihood models which describe the probability of observing a data point as a function of the model parameters. Although these two components enable data and prior knowledge to improve the human model online, likelihood models are difficult to specify and the priors may be incorrect.

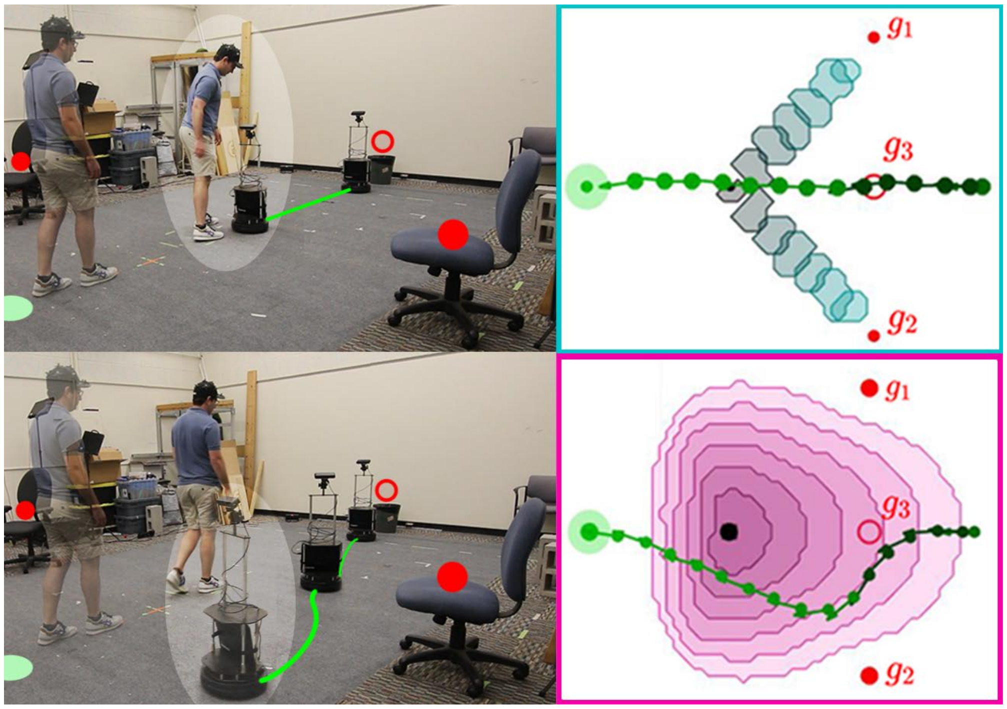


Figure 6.1: When intent-driven human models are misspecified in Bayesian predictors, robots confidently plan unsafe motions (top). Our approach (bottom) trusts the intent-driven model only to remove completely *unlikely* human actions, resulting in safer robot plans despite a misspecified model. (not depicted here) When planning using the worst-case predictor, the robot has to leave the environment entirely to avoid the predicted human state.

In this work we seek an approach which bridges robust control and intent-driven predictors: a predictor which is more robust to misspecified models and priors, but still able to leverage human data online to safely reduce conservatism. Our key idea is to compute a *restricted* forward reachable set by *trusting the intent-driven model to tell us only what is completely unlikely*. However, unlike intent-driven predictors, we will not rely on the exact probability of each action under our model during prediction. Rather, we use the decision-making model and the belief to divide the set of human actions in two disjoint sets of likely and unlikely actions. We then predict human motion by treating all sufficiently likely actions as equally probable, much like in the full forward reachable set. Using this restricted control set results in a prediction problem which can be readily formalized and solved through existing robust control methods and tools [157, 52]. We utilize Hamilton-Jacobi (HJ) reachability analysis [158, 150] which is a method for guaranteeing safety for continuous-time, nonlinear dynamical systems. Finally, to properly restrict the set of human controls based on the intent-driven model and belief over model parameters, we augment the state space with the belief. Since the belief encodes the likelihood of human actions given the history of human actions, this explicit belief tracking allows us to compute the likely actions at any future state.

To summarize, our key contributions are:

- a robust control framework for human motion prediction which provides robustness against misspecified models and model parameter priors;
- a comparison of our approach to forward reachable set and stochastic predictors for static and time-varying human intent models and in three pedestrian scenarios where the belief over the human intent changes online;
- a demonstration of our prediction approach in hardware.

6.1 Problem Setup

We consider a robot operating in a shared workspace with a human. The robot needs to predict the human's motion[1] and plan a collision-free path around the human to reach the goal as efficiently and safely as possible. To describe the motion of the robot and the human, we model both as dynamical systems. Let the state of the human and the robot be $x_\mathrm{H} \in \mathbb{R}^{n_\mathrm{H}}$ and $x_\mathrm{R} \in \mathbb{R}^{n_\mathrm{R}}$ respectively. The evolution of these states over time can be described by ordinary differential equations

$$\dot{x}_\mathrm{H} = f_\mathrm{H}(x_\mathrm{H}, u_\mathrm{H}), \;\; \dot{x}_\mathrm{R} = f_\mathrm{R}(x_\mathrm{R}, u_\mathrm{R}), \tag{6.1}$$

where the human and robot's controls are $u_\mathrm{H} \in \mathbb{R}^{m_\mathrm{H}}$ and $u_\mathrm{R} \in \mathbb{R}^{m_\mathrm{R}}$ respectively.

The robot's goal is to plan a control trajectory $u_\mathrm{R}(t), t \in [0, \bar{T}]$ such that it does not collide with the human or any (known) static obstacles, and reaches its goal g_R by $\bar{T}$. In this work, we will solve this planning problem in a receding horizon fashion. However, the future states of the human are not known *a priori*, and thus the robot must predict future human motion in order to plan collision-free trajectories.

Throughout this paper, we will focus on contrasting the intent-driven and full forward reachable set predictor with our novel predictor. However, all prediction schemes ultimately produce a set of sufficiently likely states (forward in time until the prediction horizon, N) that the robot uses for collision checking. We define the set of likely human states at some future time, $t + \tau$ as: $\mathcal{K}^t(\tau), \; \forall \tau \in [0, N]$.

Running example: *We now introduce a running example for illustration purposes throughout the paper. Consider a mobile robot that needs to navigate to a goal position $g_\mathrm{R} \in \mathbb{R}^2$ in a room where a person is walking. Since the human is a pedestrian in this scenario, we use a planar human model with state $x_\mathrm{H} = [h_x, h_y]$ and dynamics $\dot{x}_H = [v_H cos(u_\mathrm{H}), v_H sin(u_\mathrm{H})]$. We model the human as moving at a fixed speed $v_\mathrm{H} \approx 0.6 m/s$ and controlling their heading angle $u_\mathrm{H} \in [-\pi, \pi]$. Our mobile robot is modeled as a 3D system with state given by its position and heading $x_\mathrm{R} = [s_x, s_y, \phi]$, and speed and angular speed as the control $u_\mathrm{R} = [v_R, \omega]$, and dynamics*

[1]We assume that the robot and human states can be accurately sensed.

$\dot{x}_R = [v_R \cos\phi, v_R \sin\phi, \omega]$. *The robot control inputs are constrained between* $[0, 0.6]m/s$ *and* $[-1.1, 1.1]rad/s$ *respectively.*

6.2 Background: Robust vs. Intent Prediction

In this work, we aim to unify ideas from the robust control with the intent-driven prediction so we start with a brief background on both. In each section, we refer interested readers to more comprehensive resources on each approach.

6.2.1 Robust Control Prediction

The most conservative prediction of human motion is the set of *all* states the human could reach in a time horizon. Let t be the current real time and $\tau \in [0, N]$ be a future time used by the predictor. Also, let $\xi(x_{\mathrm{H}}^0, \tau, u_{\mathrm{H}}(\cdot)) := x_{\mathrm{H}}^\tau$ denote the human state starting from the current state $x_{\mathrm{H}}^0 := x_{\mathrm{H}}^t$ at time 0 and applying control u_{H} for a duration of τ. The *full Forward Reachable Set (FRS)* is defined as:

$$\mathcal{K}_{FRS}^t(\tau) := \{x_{\mathrm{H}}^\tau : \exists u_{\mathrm{H}}(\cdot), x_{\mathrm{H}}^\tau = \xi(x_{\mathrm{H}}^0, \tau, u_{\mathrm{H}}(\cdot))\} \tag{6.2}$$

In other words, if the human is in state x_{H}^0, then they are predicted to reach any state x_{H}^τ in τ time if that state is reachable through some control signal $u_{\mathrm{H}}(\cdot)$.

In general there are many techniques for computing these sets [52, 67, 157], but in this work we use Hamilton-Jacobi (HJ) reachability analysis [150, 158]. In HJ reachability, the computation of the FRS is formulated as a dynamic programming problem which ultimately requires solving for the value function $V(\tau, x_{\mathrm{H}})$ in the following initial value Hamilton Jacobi-Bellman PDE (HJB-PDE):

$$\begin{aligned} \frac{\partial V(\tau, x_{\mathrm{H}})}{\partial \tau} + \max_{u_{\mathrm{H}} \in} \nabla_{x_{\mathrm{H}}} V(\tau, x_{\mathrm{H}}) \cdot f(x_{\mathrm{H}}, u_{\mathrm{H}}) = 0 \\ V(0, x_{\mathrm{H}}) = (x_{\mathrm{H}}), \end{aligned} \tag{6.3}$$

where $\tau \geq 0$. The function (x_{H}) is the implicit surface function representing the initial set of states that the human occupies $\mathcal{L} = \{x_{\mathrm{H}} : (x_{\mathrm{H}}) \leq 0\}$. Note that this equation is the continuous-time analogue of the discrete-time Bellman equation. The maximization over the human's control, $u_{\mathrm{H}} \in$, encodes the effect of the human dynamics and control on the value, which lies in the set of all possible controls. Note that since this optimization considers all controls, the predictions will include all possible states the human could reach, thereby resulting in the safe but oftentimes overly conservative predictions. Once the value function $V(\tau, x_{\mathrm{H}})$ is computed, the FRS predictions are given by the sub-zero level set $\mathcal{K}_{FRS}^t(\tau) = \{x_{\mathrm{H}} : V(\tau, x_{\mathrm{H}}) \leq 0\}$. For more details on the HJB-PDE, please refer to [158].

6.2.2 Intent-driven Bayesian Prediction

Unlike the robust predictor, the intent-driven Bayesian predictor couples a structured model of how the human chooses their actions with the dynamics model. In general, constructing a human decision-making model for a robotic application is a particularly difficult modeling challenge and many approaches exist in the literature (see [183]). In this work, we consider stochastic control policies that are parameterized by a discrete random variable θ^t where Λ is the set of all values that θ^t can take. The human's control policy can be described by the probability density function $u_{\mathrm{H}}^t \sim p(u_{\mathrm{H}}^t \mid x_{\mathrm{H}}^t; \theta^t)$. Here, θ^t can represent many different aspects of human decision-making, including what goal locations they are moving towards [238] or even the kind of visual cues they pay attention to in a scene [117]. We refer to these aspects of human decision-making as the human's *intent*. Furthermore, we use the superscript t on the parameter to denote that the value of the human parameter can be time-varying. This allows the human model to encode how the human's intent changes over time; for example, if a person changes the goal they are moving towards in a room.

In general, the specific choice of parameterization is often highly problem specific and can be hand-designed or learned from prior data [238, 70]. Regardless of the specific parameterization, in practice, the true value of θ^t is frequently unknown beforehand and instead can be estimated from the measurements of the true human behavior. Thus, at any time t, the robot additionally maintains a belief distribution $b^t(\theta^t)$ over the model parameters, which allows it to estimate the human's intent online via a Bayesian update:

$$b_+^t(\theta^t \mid u_{\mathrm{H}}^t, x_{\mathrm{H}}^t) = \frac{P(u_{\mathrm{H}}^t \mid x_{\mathrm{H}}^t; \theta^t) b^t(\theta^t)}{\sum_{\bar{\theta} \in \Lambda} P(u_{\mathrm{H}}^t \mid x_{\mathrm{H}}^t; \bar{\theta}) b^t(\bar{\theta})} \tag{6.4}$$

Running example: *The robot has uncertainty about the human's goal location. Let the human parameter $\theta^t \in \Theta = \{g_1, g_2\}$ take two values which indicates which goal location the human moving towards. The human decision-making model at any state and for a particular goal is given by a Gaussian distribution over the heading angle with mean pointing in the goal direction and a variance representing uncertainty in the human action:*

$$p(u_{\mathrm{H}}^t \mid x_{\mathrm{H}}^t; \theta^t) = \begin{cases} \mathcal{N}(\mu_1, \sigma_1^2), & \textit{if } \theta^t = g_1 \\ \mathcal{N}(\mu_2, \sigma_2^2), & \textit{if } \theta^t = g_2 \end{cases}, \tag{6.5}$$

where $\mu_i = \tan^{-1}\left(\frac{g_i(y) - h_y^t}{g_i(x) - h_x^t}\right)$ and $\sigma_i = \pi/4$ for $i \in \{1, 2\}$. Here, $(g_i(x), g_i(y))$ represents the position of goal g_i.

At prediction time, the stochastic nature of the human decision-making model and the belief over the parameters is naturally converted into state *distributions* (instead of deterministic sets) forward in time. Note that typically, these predictors use a temporally and spatially discretized form of the dynamics by integrating f_{H} over a fixed time interval δt. Controls are often discretized too and assumed to be held fixed during δt. This results in the predictor maintaining and updating discrete distributions over the human state space. Given the current real time t, we will denote a future time discrete timestep by $k \in \{0, 1, \dots, \frac{H}{\delta t}\}$.

Suppose the current state of the human at the start of the prediction horizon is $x_{\mathrm{H}}^0 := x_{\mathrm{H}}^t$ and the current belief is $b^0(\lambda^0) := b^t(\lambda^t)$. Assume the human is at x_{H}^k at some future time k. Combining the dynamics and human policy, the human's state distribution at the next timestep $k+1$ is

$$P(x_{\mathrm{H}}^{k+1} \mid x_{\mathrm{H}}^k; \theta^k) = \sum_{u_{\mathrm{H}}^k} P(x_{\mathrm{H}}^{k+1} \mid x_{\mathrm{H}}^k, u_{\mathrm{H}}^k) P(u_{\mathrm{H}}^k \mid x_{\mathrm{H}}^k; \theta^k).$$

This equation can be applied recursively to compute $P(x_{\mathrm{H}}^{k+1} \mid x_{\mathrm{H}}^0; \theta^{0:k})$ starting from $k = 0$. Marginalizing over all sequences of values that the human parameter θ could take, $\mathcal{S}_k$, where $|\mathcal{S}_k| = |\theta|^k$, we get the overall distribution over the human state at future time step $k+1$: $P(x_{\mathrm{H}}^{k+1} \mid x_{\mathrm{H}}^0)$.
Here, the probability of the parameter sequence has to be set in the model and is generally defined by [2]:

$$P(\theta^{0:k} \mid x_{\mathrm{H}}^0) = \Big(\prod_{m=1}^{k} P(\theta^m \mid \theta^{m-1})\Big) b^0(\theta^0)$$

.

Importantly, at planning time, the robot must decide which predicted states are sufficiently likely to warrant avoiding. A strict notion of safety requires the robot to avoid all states whose probability is > 0. While safe (and equivalent to the full FRS), this choice of states does not leverage the data encoded through the belief or the human decision-making model. To reduce the volume of this set in a way commensurate with human decision-making, choosing a nonzero probability threshold is desirable and reveals a significantly smaller set of states that aligns with the model. Thus, the ultimate predicted set of human states that the robot must avoid at planning time is:

$$\mathcal{K}_\epsilon^t(k) = \{x_{\mathrm{H}}^k : P(x_{\mathrm{H}}^k \mid x_{\mathrm{H}}^0) > \epsilon\}, \forall k \in \{0, \dots, \frac{N}{\delta t}\} \tag{6.6}$$

where $\epsilon \geq 0$ is a safety threshold and a design parameter.

6.3 A Robust-Control Framework for Intent-Driven Human Prediction

Our key idea in this paper is to compute a *restricted* forward reachable set by trusting the intent-driven model to infer only what is *completely unlikely*. After using the intent-driven

[2]In the case of static latent parameters, the summation simplifies to $P(x_{\mathrm{H}}^{k+1} \mid x_{\mathrm{H}}^0) = \sum_{\theta \in \Lambda} P(x_{\mathrm{H}}^{k+1} \mid x_{\mathrm{H}}^0; \theta) b^0(\theta)$.

model to prune away sufficiently unlikely actions, our robust predictor will safeguard against all sufficiently likely actions equally, much like in the full forward reachable set. The main question becomes how to perform this control-set pruning in a principled way over the prediction horizon.

One simple way of choosing this set is as follows. At the beginning of the prediction horizon, let the human state be $x_{\mathrm{H}}^0 := x_{\mathrm{H}}^t$ and the current belief be $b^0 := b^t$. We can form a new distribution over the human's controls at the first time step by marginalizing out the latent model parameters, given the initial belief we have over those parameters: $p(u_{\mathrm{H}} \mid x_{\mathrm{H}}) = \sum_{\theta\in\Lambda} p(u_{\mathrm{H}} \mid x_{\mathrm{H}}; \theta) b^0(\theta)$. Then, we can choose the set of human actions to be those for which this marginalized initial likelihood is above a threshold: $\boldsymbol{\mathcal{U}}(x_{\mathrm{H}}) = \{u_{\mathrm{H}} : p(u_{\mathrm{H}} \mid x_{\mathrm{H}}) \geq \delta\}$. This leads to a set of reachable states for $t = 1$. To obtain the set of states at $t = 2$, it is tempting to follow the same process, restricting the set of future actions based on b^0. Unfortunately, this would (accidentally) model that the human is "resampling" their intent from this initial distribution independently at every step. It would disregard that a human's second action will be *consistent* with their first, with the intent only changing according to the dynamics of θ. Thus, we must enforce that the human control from a state x_{H} at $t = 2$ is not only consistent with the initial belief, but also with the control that took them to state x_{H}.

To properly restrict the set of feasible controls over the prediction horizon, we need to take into account how the likelihood of any future control depends on the past sequence of human controls. The belief precisely encodes this likelihood given the past sequence of human controls through the Bayesian update from Eq. 6.4. Thus, our predictor explicitly tracks the updated belief as it makes predictions, rather than just the updated state, and restricts future actions based on future beliefs (see left of Fig. 6.3 for intuitive depiction). Let this joint state space be denoted by $x^t := \begin{bmatrix} x_{\mathrm{H}}^t & b^t(\lambda^t = \lambda_1) & \dots & b^t(\lambda^t = \lambda_{|\Lambda|}) \end{bmatrix}$.

When predicting using this state space, to simultaneously predict the possible future beliefs over θ^t and corresponding likely human states, we consider the joint dynamics:

$$\dot{x}^t = \begin{bmatrix} \dot{x}_H^t & \dot{b}^t(\theta^t = \theta_1) & \dots & \dot{b}^t(\theta^t = \theta_{|\Lambda|}) \end{bmatrix}, \tag{6.7}$$

where $\dot{x}^t := f(x^t, u_{\mathrm{H}}^t)$. The continuous evolution of the belief $b^t(\theta^t)$ can be described by:

$$\dot{b}^t(\theta^t) = \gamma\Big(b_+^t(\theta^t \mid u_{\mathrm{H}}^t, x_{\mathrm{H}}^t) - b^t(\theta^t)\Big) + \big(b^t(\theta^t)\big) \tag{6.8}$$

for any specific value of θ^t. Here, the function $(\cdot)$ represents the intrinsic changes in the human intent, whereas the other component captures the Bayesian change in $b^t(\theta^t)$ due to the observation u_{H}^t. Note that the time derivative in (6.8) is pointwise in the space of all θ's. Typically, the Bayesian update is performed in discrete time when the new observations are received. However, to unify this with continuous-time robust controls tools, in this work, we reason about continuous changes in $b^t(\theta^t)$. Intuitively, to relate the continuous-time Bayesian update to the discrete-time version, γ in (6.8) can be thought of as the observation frequency. Indeed, as $\gamma \uparrow \infty$, i.e., observations are received continuously,

$b^t(\theta^t)$ instantaneously changes to $b^t_+(\theta^t \mid u^t_{\mathrm{H}}, x^t_{\mathrm{H}})$. On the other hand, as $\gamma \downarrow 0$, i.e., no observation are received, the Bayesian update does not play a role in the dynamics of $b^t(\theta^t)$. For a detailed derivation of continuous dynamics, we refer the interested readers to Sec. 6.7.

Now that we are able to track the evolution of the robot's belief and the human's physical states, we can prune unlikely human actions by combining the intent-driven model and the *predicted* belief over the human model parameters. For some future time $\tau \in [0, N]$, the marginalized human action distribution at joint state x^τ is given by

$$p(u^\tau_{\mathrm{H}} \mid x^\tau) = \sum_{\theta \in \Lambda} p(u^\tau_{\mathrm{H}} \mid x^\tau_{\mathrm{H}}; \theta) b^\tau(\theta). \tag{6.9}$$

Very importantly, note that this set is joint state dependent, and therefore *belief*-dependent. This allows us to prune away the sufficiently unlikely actions by removing actions which are not assigned sufficient probability under the future predicted belief (and not just the initial belief):

$$u^\tau_{\mathrm{H}} \in \mathcal{U}_{\mathrm{H}}(z^\tau), \quad \mathcal{U}_{\mathrm{H}}(z^\tau) = \{u^\tau_{\mathrm{H}} : p(u^\tau_{\mathrm{H}} \mid x^\tau) \geq \delta\} \tag{6.10}$$

where $p(u_{\mathrm{H}} \mid x)$ is computed as in Eq. (6.9) and δ is a threshold that partitions the actions into likely and unlikely.

Running example: *Consider the case when the intrinsic behavior of the human does not change over time, i.e.,* $(b^t(\theta^t)) = 0$*, meaning the human has a fixed goal they are moving to. Since* θ *takes only two possible values, the joint state space is three dimensional. In particular,* $x = \begin{bmatrix} h_x & h_y & p_1 \end{bmatrix}$*, where* $p_1 := b^t(\theta^t = g_1)$ *and* $b^t(\theta^t = g_2)$ *is given by* $\left(1 - b^t(\theta^t = g_1)\right)$ *so we do not need to explicitly maintain it as a state. The state-dependent control distribution is* $p(u_{\mathrm{H}} \mid x) = p_1 \mathcal{N}(\mu_1, \sigma_1^2) + (1 - p_1)\mathcal{N}(\mu_2, \sigma_2^2)$ *and can be used to compute the set of allowable controls for different values of* δ *via Eq.* (6.10)*. Note Fig.* 6.2 *where the top-left inset figures show the allowable controls for* $x = (0, 0)$ *and two different belief states* $b^0(\theta = g_1) = 0.5$ *and* $b^0(\theta = g_1) = 0.9$ *for three different* δ *thresholds.*

6.3.1 Using HJ-Reachability for Prediction

Using a control set rather than a distribution results in a prediction problem which can be readily solved using the HJB-PDE formulation in Section 6.2.1. At any real time t, given the current state of the human and the current belief over the model parameters, we can construct the joint state at the beginning of the prediction horizon $x^0 := x^t$. Using this initial state and the thresholded control policy from (6.9), we are interested in computing the following set:

$$\mathrm{FRS}(\tau) := \{x^\tau : \exists u_{\mathrm{H}}(\cdot) \in \mathcal{U}_{\mathrm{H}}(x), x^\tau = \xi(x^0, \tau, u_{\mathrm{H}}(\cdot))\}, \tag{6.11}$$

where $\tau \in [0, N]$. Intuitively, $\mathrm{FRS}(\tau)$ represents all possible states of the joint system, i.e., all possible human states and beliefs over θ, that are reachable under the dynamics in

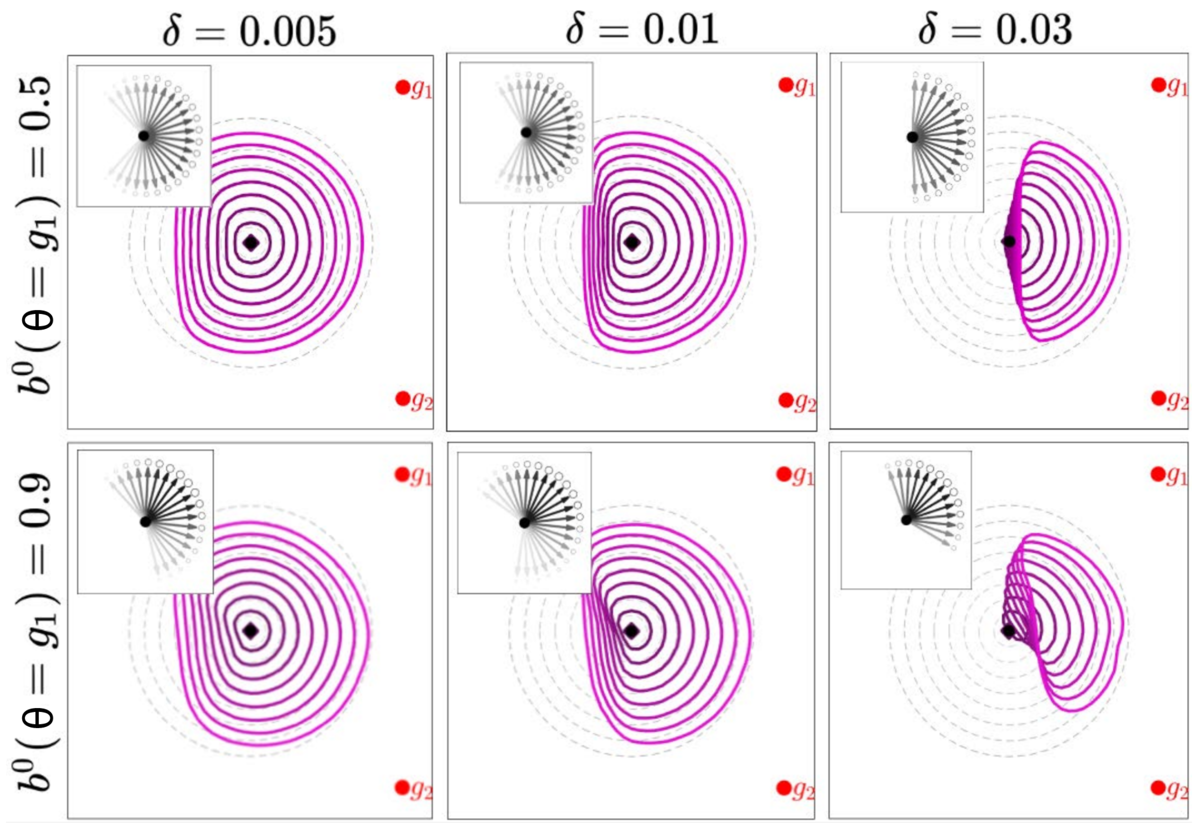


Figure 6.2: Effect of the belief and the δ-threshold on the admissible set of controls (shown in upper-left inset) and the overall predictions (shown in pink) for 3 seconds into the future.

(6.7) for some sequence of human actions. We refer to this set as the *Belief Augmented Forward Reachable Set (BA-FRS)* from here on. Much like the computation of the full forward reachable set from Sec. 6.2.1, we can leverage the same tools from HJ-Reachability to compute the BA-FRS where x_{H} is replaced with x and instead of optimizing over all controls , we use the restricted set of controls (x) instead.

After solving the dynamic programming problem to obtain the BA-FRS from (6.11), our predictions include not only the physical locations of the human but also the corresponding future beliefs. However, for motion planning, the robot needs to collision-check against a set of physical states the human could occupy. We obtain this set by projecting FRS(τ) on the human state space via $\mathcal{K}^t_\delta(\tau) = \bigcup_{x^\tau \in \mathrm{FRS}(\tau)} \Pi(x^\tau), \quad \forall \tau \in [0, N]$ where $\Pi(x)$ is the physical state component of x.

Running example: *Our starting set of states, $\mathcal{L}$, is a small ball at the joint starting state* $z^0 = [0, 0, 0.5]$*, shown in grey in Fig. 6.3. Consider how the state and belief can change in a small (*$\delta t = 0.4668$*) timestep after observing the person moving towards goal 1 via* $u_{\mathrm{H}} = \pi/4$*. Since this action is highly likely under the model where* $\theta = g_1$*, then the next joint state will have the person not only moved physically in that direction, but the posterior will have increased probability*

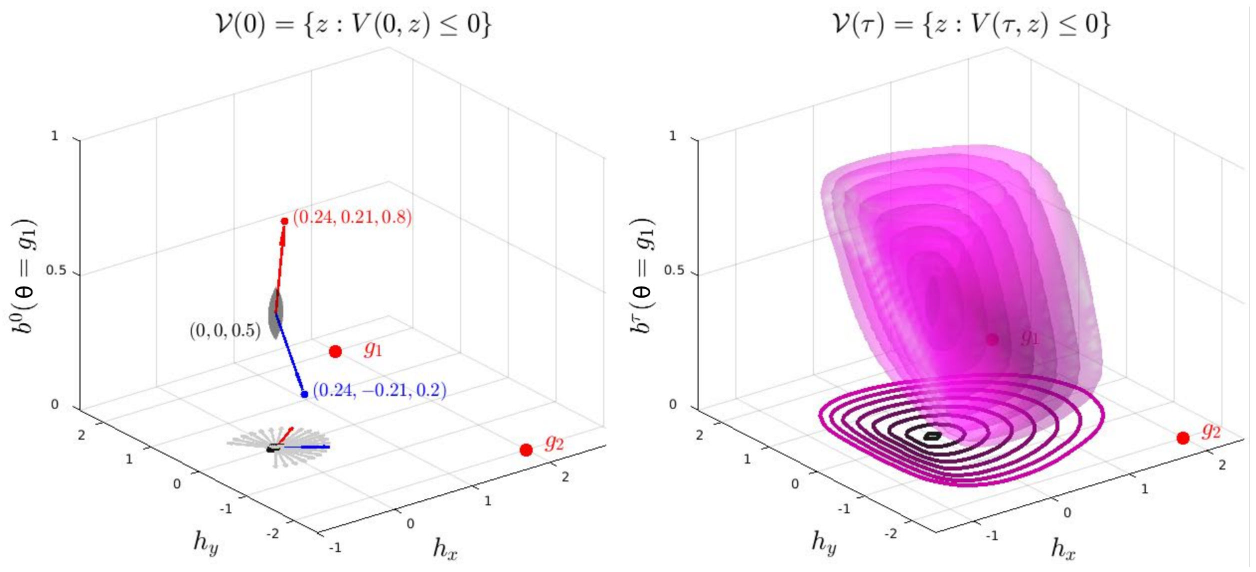


Figure 6.3: (left) Initial set in x-space (in grey). Likely control distribution for $\delta = 0.01$ shown projected in $h_x - h_y$ plane. Comparisons of the resulting joint state if the human moves directly towards g_1 (in red) versus towards g_2 (in blue). (right) 4 seconds BA-FRS and its projection into x_H-space.

mass on $b(\lambda = g_1)$. Similarly, this probability decreases if the human moves to g_2. After solving for $V(\tau, x)$ via (6.3)*, we take the sub-zero level set to retrieve the joint state predictions (Fig. 6.3, right), and the predictions $\mathcal{K}_\delta^t$ after projecting onto the human's state space.*

In summary, to predict the human's motion, our predictor optimizes the initial value HJB-PDE from (6.3) but instead of optimizing over *all* controls, our formulation modifies Eq. (6.3) to maximize over the restricted set $u_H \in (z)$ which changes based on human state, time, and belief. Ultimately, the proposed prediction framework is a *less conservative* FRS, but a *more conservative* intent-driven predictor. This has two advantages: (1) when the intent-driven model is correct, it computes an under-approximation of the full FRS to reduce conservatism in a principled way, and (2) when the model is incorrect, we can be more robust to such inaccuracies since the predictions no longer rely on the exact action probabilities. We discuss this further in Sec. 6.4.

6.4 Prediction Comparisons

We now compare our predictor with the intent-driven Bayesian predictor and the full FRS when (1) the human intent is static, (2) the human intent is time-varying, and (3) the human moves in unmodelled ways over time.

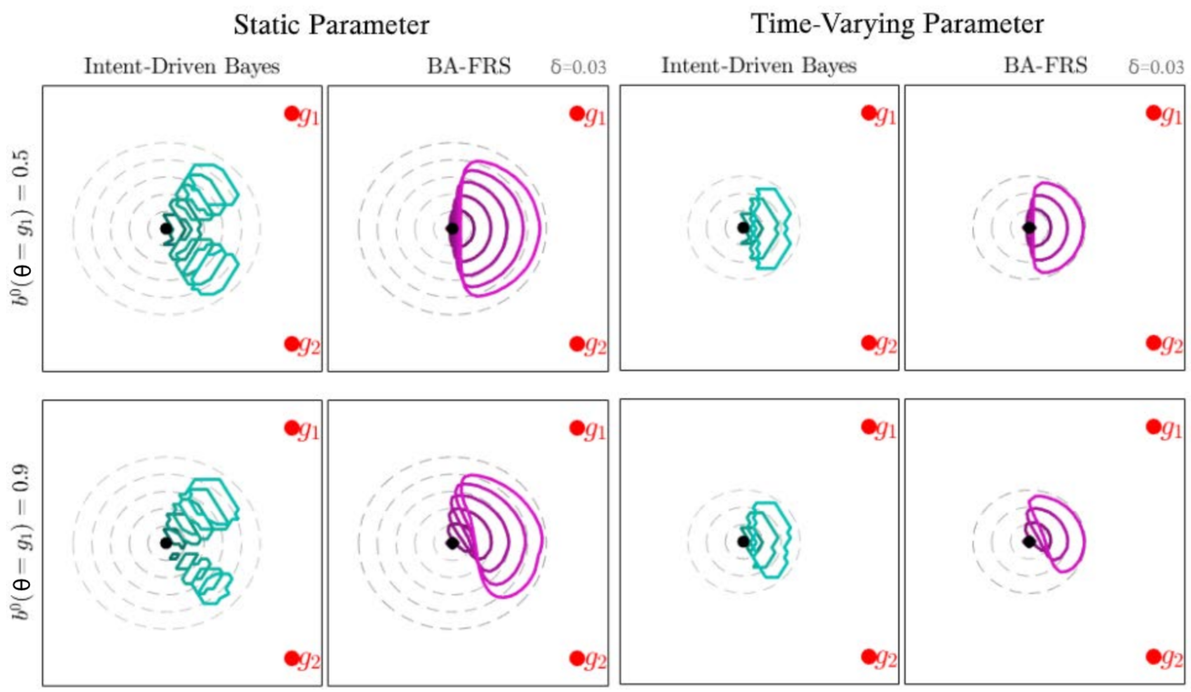


Figure 6.4: Comparisons of Bayesian and BA-FRS predictions for static vs. time-varying human intent. Dashed lines are the full FRS. Predictions are for 2 seconds for the static parameter and 1.8 seconds for time-varying. For the Bayesian predictor, we choose ϵ to capture the $(1 - \delta)$ most likely states.

6.4.1 Static parameter

One simple but common predictive model of human behavior assumes that the human's intent (and thus model parameter θ) is static. In our running example, this means the person has a fixed goal location they are moving towards and they will not change their mind. Mathematically, in the intent-driven Bayesian predictor, this is represented via the θ transition distribution $P(\theta^{k+1} \mid \theta^k) = \mathbb{1}_{\{\theta^{k+1}=\theta^k\}}$ where $\mathbb{1}$ represents the indicator function. In the BA-FRS predictor it means $k(b^t(\theta^t)) = 0$ in the distribution dynamics.

In the left block images in Fig. 6.4, we see a snapshot of predictions generated by the intent-driven predictor and the BA-FRS forward in time for 2 seconds ($N = 18$). The full forward reachable set is visualized as a series of concentric dashed grey circles. The top row represents a uniform belief over the two goals, while the bottom row represents a high belief on goal 1 (g_1). As expected, both the intent-driven predictor and the BA-FRS are far less conservative than the full FRS. Furthermore, the set of sufficiently likely states predicted by the intent-driven Bayesian predictor is always contained within the BA-FRS. Consequently, when the belief over θ is confident that the human is moving towards g_1 (see bottom row of Fig. 6.4), then the BA-FRS allows us to compute an approximation of the stochastic predictor.

6.4.2 Time-varying parameter

A more complex model of human intent allows it to vary over time. To encode this time-varying intent in the predictor, we need a model of how the human chooses the next value of λ^t. In our running example, let a simple model for how the person changes their behavior to be:

$$P(\theta^{k+1} \mid \theta^k) = \begin{cases} \alpha + (1-\alpha)\cdot\Delta(\theta^k) & \text{if } \theta^{k+1} = \theta^k \\ (1-\alpha)\cdot\Delta(\theta^{k+1}) & \text{if } \theta^{k+1} \neq \theta^k \end{cases} \tag{6.12}$$

where α is a known model parameterwhich governs how likely the person is to change their intent and Δ is a known discrete distribution over the model parameters. This model encodes that if the person was moving towards $\theta^k = g_1$ at the previous timestep k, they are likely to continue to walk to g_1 at the next timestep with probability $\alpha + (1-\alpha)\cdot\Delta(\theta^k = g_1)$, or they can switch to $\theta^{k+1} = g_2$ at the next timestep with probability $(1-\alpha)\cdot\Delta(\theta^{k+1} = g_2)$. In the BA-FRS predictor, this time-varying intent model is encoded via the distribution dynamics: $k(b^t(\theta^t)) = \alpha b^t(\theta^t) + (1-\alpha)\cdot\Delta(\theta^t) - b^t(\theta^t)$.

In the right block of images in Fig. 6.4, we see a snapshot of predictions when the latent parameter is time-varying forward in time for 1.8 seconds ($H = 11$). Note that when the parameter is time-varying, the computational complexity of the intent-driven Bayes predictor exponentially increases in the size of the prediction horizon, $|\Lambda|^N$, due to the necessity of tracking all sequences of values that θ can take over time. In practice, prediction was computationally prohibitive for horizons greater than 1.8 seconds. In contrast, the BA-FRS computation grows linearly in the length of the prediction horizon, but exponentially in the number of parameter values due to the addition of the belief in the state. Thus, for time-varying parameters which take a few values and for longer prediction horizons, our prediction method can be particularly suitable for getting an approximation of Bayes predictor at a lower computational complexity.

When θ is static, then the intent-driven predictor with a high belief over g_1 deems moving directly towards g_2 to be highly unlikely. However, when θ is time-varying, the human can "switch" which goal they are moving towards, thereby making states in the direction of g_2 somewhat likely as well. For the BA-FRS, even though directly moving towards g_2 is unlikely under the intent-driven model and belief, the BA-FRS realizes that moving *away* from g_1 is *still likely enough*. Consequently, the predicted BA-FRS mass moves in the direction of g_2 over time, in the case of both static and time-varying θ, allowing us to be robust to suboptimal human trajectories as discussed in the next section.

6.4.3 Online updates & robustness to misspecified models

Ultimately, both the intent-driven Bayesian predictor and the BA-FRS will update the belief over the human parameters online based on how the person moves. Here we simulate three scenarios–one where the person takes a path well-modelled by the intent-driven model and two where the person behaves in an unmodelled way–and discuss how our framework ensures robustness in situations like these.

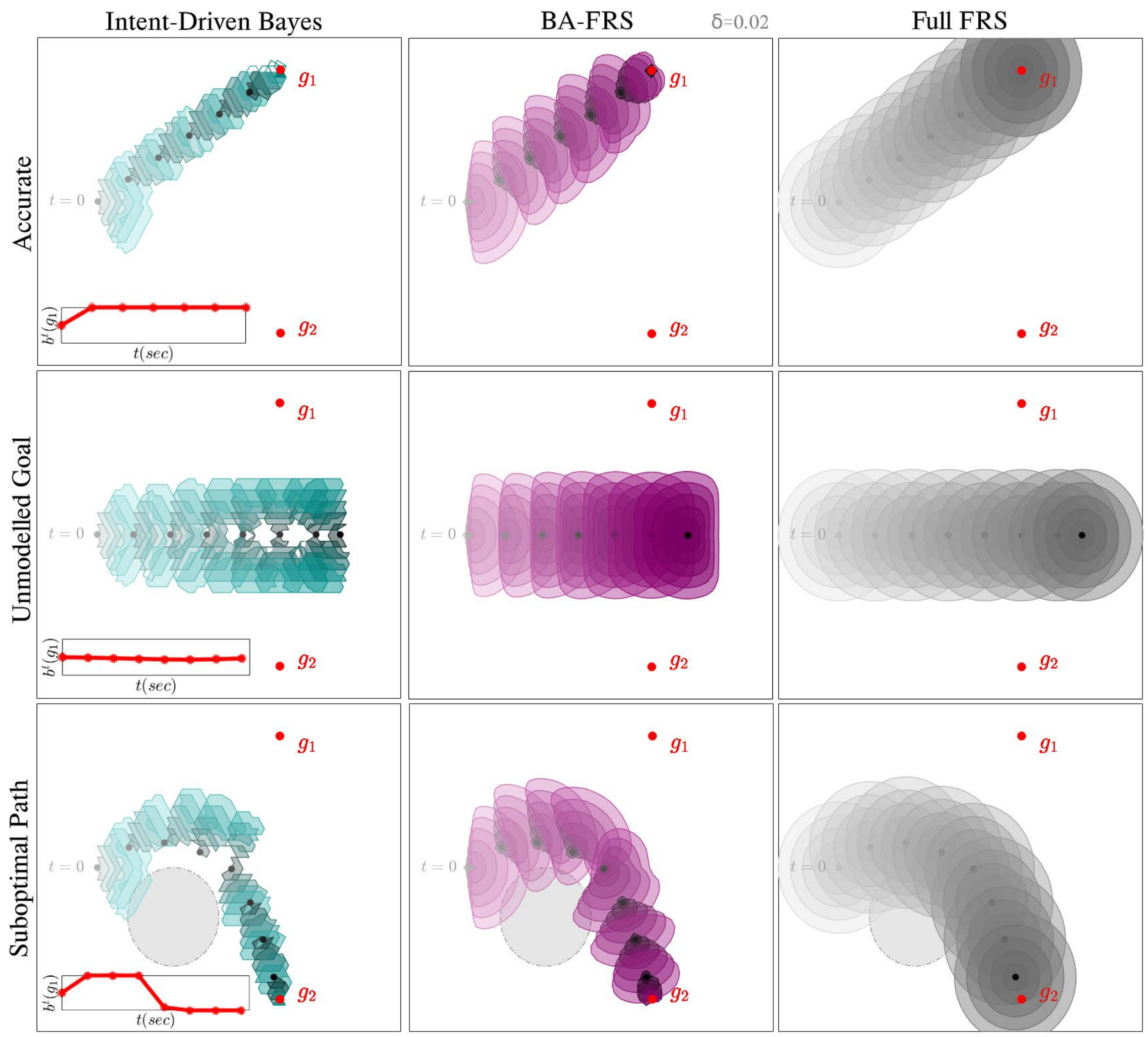


Figure 6.5: Comparison of the intent-driven Bayesian, our BA-FRS, and the full FRS predictions for three scenarios. In the first row the human moves towards one of the modelled goals. In the middle the human moves towards an *unmodelled* goal. In the last row the human is moving towards a modelled goal (g_2) but they take a suboptimal path under our model because they are avoiding an *unmodelled* obstacle on the ground (shown in grey circle). The belief over g_1 is visualized over time in the lower-left inset plot.

In all examples, the predictors begin with a uniform prior over the two goals, use a static model of human intent, and the BA-FRS uses a $\delta = 0.02$. In the top row of Fig. 6.5 the human has a fixed intent to move towards the upper goal 1 (g_1). In this scenario, the intent-driven model is correctly specified and as the person moves towards g_1, the belief over g_1 increases and the Bayesian predictions focus towards this goal. Our

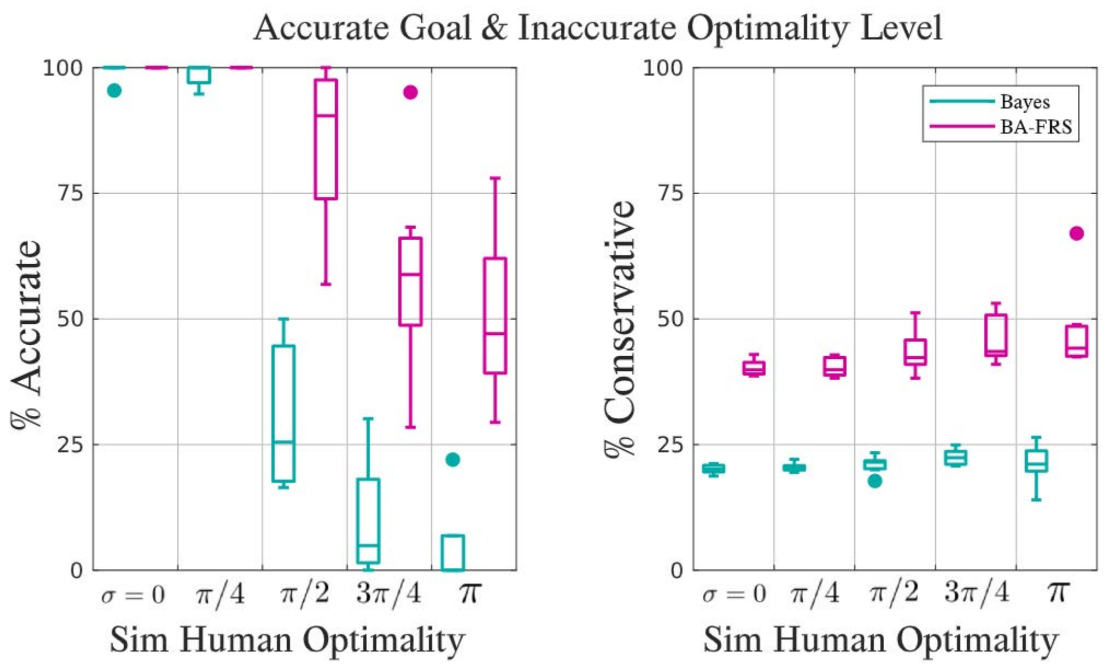


Figure 6.6: Simulated human moves to modelled g_1 but with varied optimality from $\sigma = 0$ (optimal) to π (random). Both predictors use a fixed $\sigma = \pi/4$.

BA-FRS performs similarly since it too performs the belief update over time. However, since the BA-FRS explicitly tracks the evolution of the belief in the future during prediction, the sets include more states even in the direction of g_2. This is because the predictions are safeguarding against slightly suboptimal actions which are still likely enough under the model and would lead to the belief over g_1 *decreasing* in the future. Nevertheless, the BA-FRS takes up significantly less volume than the full FRS, thereby reducing overall conservativism.

The second and third rows demonstrate two human behaviors that are unmodelled – a scenario where the human is actually walking towards a third unmodelled goal in between g_1 and g_2 and a scenario where the human takes a seemingly suboptimal path to g_2 due to an unmodelled obstacle. In the later scenario, the belief over g_1 sharply increases as the person moves around the unknown obstacle. This results in the Bayesian predictor being overly optimistic, and it places most probability mass on states that are in the direction of g_1. In contrast, our predictor remains cautiously conservative because (1) it is safeguarding against the slightly suboptimal but still sufficiently likely actions and (2) it is evolving the belief during the prediction horizon. In fact, the true sequence of human states and actions lies within the predictions of the BA-FRS, ensuring that a robot which relies on these predictions will in fact avoid the states that the human eventually occupies. We discuss the middle scenario in greater detail in Sec. 6.5.

We conducted a series of additional experiments with simulated human trajectories to

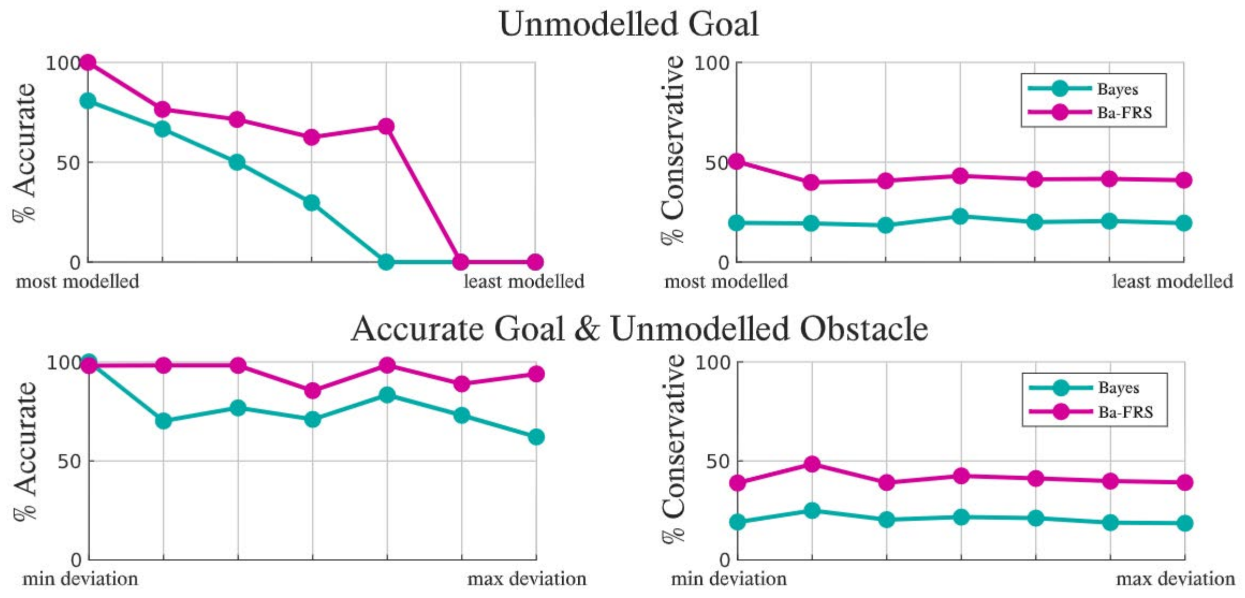


Figure 6.7: (top) Simulated human moves to one of 7 *unmodelled goals* in an optimal trajectory. Results are in increasing misspecification of the goal. (bottom) Simulated human moves to modelled goal g_2 but has their straight-line path obstructed by an *unmodelled obstacle*. Results from 7 unmodelled obstacles are shown in increasing deviation from the straight trajectory.

systematically analyze the three misspecification types: (1) accurate goal but inaccurate optimality level, (2) unmodelled goal, (3) accurate goal but unmodelled obstacle. We compare different predictors for prediction accuracy and conservatism. A predictor is considered accurate at a particular time step if the true future human states lies within the predictions for the entire prediction horizon. Conservatism is measured by computing the percent volume of the full FRS that the predicted states occupy. Both the accuracy and conservatism metrics are computed at each time step and averaged over the horizon $[0, \bar{T}]$. Note that the full FRS always achieves 100% accuracy but also 100% conservatism. These metrics provide us a proxy for the prediction's effect on the safety and efficiency of the robot's plan; ideally, a predictor should have high accuracy and low conservatism over the entire human trajectory. In all experiments, the Bayesian and BA-FRS predictors modelled two goals and used a fixed $\sigma = \pi/4$ in the action model described in the running example from 6.2.2 and $\delta = 0.02$.

For (1), the human was simulated as moving towards modelled goal g_1 by sampling an action $u(x) \sim \mathcal{N}(\mu_1, \sigma^2)$. To capture a range of human behavior from completely optimal to completely random, we simulated five levels of σ (depicted in Fig. 6.6). We sampled 7 random initial human states for each σ and averaged results over these trials. Fig. 6.6 shows box plots of our metrics for Bayesian and BA-FRS. Although the BA-FRS is about twice as conservative as Bayesian, it maintains a high prediction accuracy across all optimality levels, while still being far less conservative than the full FRS (BA-FRS is $\approx$ 45% of the full FRS).

For (2) and (3) we fixed the human's initial condition but varied the unmodelled goal or unmodelled obstacle. For unmodelled goals, we randomly sampled 7 unmodelled goals which were diversely spread in the (x,y)-plane. The true (unknown) human trajectory is a straight line to the unmodelled goal starting from the initial position. Fig. 6.7 (top) shows plots of the accuracy and conservatism for each of the unmodelled goals, sorted from the "most" modelled (e.g. an unmodelled goal which is nearby a modelled goal) to "least" modelled. For unmodelled obstacles, the simulated human always moved towards g_2, but their straight-line path was always obstructed by an unmodelled obstacle, forcing them to take a trajectory around the obstacle. We simulated 7 of these trajectories around various circular and rectangular-shaped obstacles. Fig. 6.7 (bottom) shows the results sorted from least deviation from straight-line trajectory to the goal to most deviation. The more irrational the human "appears" (either due to an unmodelled goal or taking a suboptimal path to the goal), the more the drop in accuracy of the Bayesian predictor, as it overrelies on the intent model to explain the human's behavior. In contrast, since BA-FRS only uses the human model to filter likely and unlikely actions, it maintains a relatively higher accuracy.

6.5 Implications for Safe Motion Planning

Consider the scenario where the actual human goal is midway between the modelled goals g_1 and g_2 (see g_3 label in Fig. 6.1 and middle row of Fig. 6.5), but the true human goal is not explicitly modelled in the intent-driven model. We will use this example in simulation and in hardware to demonstrate the challenges with over-relying on a misspecified intent-driven model. Our hardware experiments are performed on a TurtleBot 2 navigating around a human pedestrian. We measured human positions at 200Hz using a VICON motion capture system and used on-board odometry sensors for the robot state measurement. The robot is modelled via the dynamics in Sec. 6.1, its goal g_R is behind the initial state of the human (green circle in Fig. 6.1) and it uses a spline-based planner [219] to plan six-second trajectories in a receding-horizon fashion.

When the robot uses the full FRS for human motion prediction (see Fig. 6.5 for visualizations of the predictions over time), the robot plans a trajectory which deviates significantly from the ideal straight line path towards its goal and in fact forces the robot to leave the testbed[3]. In contrast, the Bayesian predictor consistently predicts that that pedestrian will walk towards one of the goals and fails to assign sufficient probability to the true human states because of its over reliance on the model. Ultimately, this leads to a collision between the human and the robot (top row Fig. 6.1). Our proposed approach does not rely heavily on the exact action probabilities, and infers that the straight line trajectory is likely enough under the pedestrian model. As a result, the robot makes a course correction early on to reach its goal without colliding with the pedestrian (bottom row Fig. 6.1).

[3]Hardware demonstration videos: https://youtu.be/uZi-zIi1S6A

6.6 Conclusion

When robots operate around humans, they often employ intent-driven models to reason about human behavior. Even though powerful, such predictors can make poor predictions when the intent-driven model is misspecified. This in turn will likely cause unsafe robot behavior. In this work, we formulated human motion prediction as a robust control problem over the set of only sufficiently likely actions, offering a bridge between conservative full forward reachable set predictors and intent-driven predictors. We demonstrated that the proposed framework provides more robust predictions when the prior is incorrect or human behavior model is misspecified, and can perform these predictions in continuous time and state using the tools developed for reachability analysis. In the future, we will scale it to higher dimensions with multiple humans, perform a user study to gauge the impact of our predictor on a humans' comfort in close navigation scenarios, and integrate it with online model confidence estimation approaches.

6.7 Derivation: Continuous-time Distribution Dynamics

Assume that every T seconds[4] we are guaranteed to receive a measurement, u_{H}. Let the current time be denoted by t and the current belief over the latent parameter θ be $b^t(\theta)$. During the time interval $[t, t+T]$ we will receive a single measurement with probability 1. For simplicity, we assume that the arrival time of the measurement is uniformly distributed in the interval $[t, t+T]$. However, depending on the measurement model, a similar derivation can be performed for other arrival time distributions as well. We want to understand how the belief $b^t(\theta)$ can change in an arbitrarily small timestep, δt, along $[t, t+T]$. As this timestep goes to zero, we will be able to recover the continuous-time dynamics of $b^t(\theta)$. Let E be a discrete random variable which takes values capturing the event that we receive a measurement within the time interval $[t, t+\delta t]$. The probability distribution of E can be written out as:

$$E = \begin{cases} e_1 = \text{measurement}, & \text{with probability } \frac{\delta t}{T} \\ e_2 = \text{no measurement}, & \text{with probability } 1 - \frac{\delta t}{T} \end{cases}$$

When a measurement is received, a Bayesian update on the belief is performed as per (6.4). When a measurement is not received, the only change in the belief is intrinsic. Let the function $(b^t(\theta))$ represent the intrinsic changes in the human behavior. Using the law of total probability:

$$\begin{aligned} b^{t+\delta t}(\theta) &= P(e_1)P^{t+\delta t}(\theta \mid x, e_1) + P(e_2)P^{t+\delta t}(\theta \mid x, e_2) \\ &= \left(\frac{\delta t}{T}\right) b^t_+(\theta \mid u, x) + \left(1 - \frac{\delta t}{T}\right)\left(b^t(\theta) + \delta t \cdot (b^t(\theta))\right). \end{aligned}$$

[4]Here, T can represent the publishing rate of the motion capture or estimator which computes the current human state (and then observation).

Rearranging some terms, we get:

$$\begin{aligned} b^{t+\delta t}(\theta) - b^t(\theta) = &\left(\frac{\delta t}{T}\right)\left(b^t_+(\theta \mid u, x) - b^t(\theta)\right) \\ &+ \delta t \cdot (b^t(\theta)) + \text{h.o.t}, \end{aligned}$$

where h.o.t includes all the terms with δt^2 in them. Taking the limit as $\delta t \to 0$ we get the time-derivative of $b^t(\theta)$: $\dot{b}^t(\theta) = \frac{1}{T}\Big(b^t_+(\theta \mid u, x) - b^t(\theta)\Big) + (b^t(\theta))$. Note that the higher order terms disappear when we take the limit of $\delta t \to 0$. We now have a form for our dynamics when our belief can change both because of a new measurement and because of the intrinsic dynamics of the human. If we let $\gamma = 1/T$ then we get the form in Equation (6.8):

$$\dot{b}^t(\theta) = \gamma\Big(b^t_+(\theta \mid u, x) - b^t(\theta)\Big) + (b^t(\theta)). \tag{6.13}$$

Part II

Formalizing Safety Analysis of Adaptive Human Models

Because humans vary in their intentions and preferences, robots must adapt their human models while interacting with people. For example, at an intersection, an autonomous car needs to infer if an oncoming human driver wants go forward or turn so it can safely make an unprotected left turn. However, an outstanding safety challenge with such online learning in human-robot interaction is quantifying *how* a human model may change in light of new data and *how long* will this change take? Here, the autonomous car should understand "When in the future will I know the human driver's intent?" so it can safely execute the unprotected left turn. The key idea in Part II is to *model the robot's learning algorithm as a dynamical system, where the state contains the estimate of the human model's parameters, and the control is the human data the robot observes*. Determining how a human model may change reduces to answering when and what states our new dynamical system can reach. Such reachability analysis can be formulated as an optimal control problem whose solution captures (1) which human data "steer" the learning system into desired states and (2) the time to learn desired human model parameters starting from an initialization. We demonstrate the utility of our analysis framework in four human-robot domains, including autonomous driving and indoor navigation.

Chapter 7

Analyzing Human Models that Adapt Online

This chapter is based on the paper "Analyzing Human Models that Adapt Online" [17] written in collaboration with Anand Siththaranjan, Claire Tomlin, and Anca Dragan.

Robots rely on predictive models of human behavior in order to plan safe and efficient motions around people. Because people vary in their intentions, preferences, and styles of behavior, these models must adapt online upon observing a specific person. For instance, a robot may use Bayesian inference to update its belief about the location a pedestrian is walking to [238, 100, 117], or use online gradient descent to update parameters corresponding to a human's proxemics preferences [19, 33] or parameters of a neural network which predicts how a human reaches for objects [53].

Enabling robots to adapt their human models online is necessary and beneficial, but it also raises safety challenges. The human model can now change significantly in light of new data–the human state/actions–that the robot observes. What the model parameters change to and how quickly they change depends both on the robot's learning algorithm and how the human actually behaves. Some parameter initializations will be conducive to better learning. Some human behaviors may be ambiguous under the model and result in slower learning. These all have significant implications for the safety and efficiency of the robot's decision-making.

Adaptive models thus raise several questions. First, what is the *worst-case time* it can take the robot to learn the correct model parameters (whatever they may be)? With this, the robot can, for instance, make contingency plans that safeguard against all intents until this worst-case time but then branch on the true parameter value afterwards. In contrast, what is the *best-case time* to learn? Here, a robot can, for example, gain insights about which locations in a room make learning from nearby humans so challenging that even in the best-case it still takes too long to estimate the human's intent. Relatedly, we can also ask what *observations* lead to the *fastest* or *slowest learning*, thereby producing legible or deceptive motions. Finally, what *initialization* should the robot use, so that we can guarantee that the robot learns the correct parameters– regardless of what they might be–in a finite number of observations? These questions amount to knowing *what* the robot can learn and in *how much time*, and thus far have been implicitly raised but received little attention.

In this work, our key idea is that we can answer these analysis questions by posing them as *reachability* queries. To achieve this, *we model the robot's learning algorithm as a* dynamical system, *where the state is the current parameter* estimate, *and the control is the human* data *the robot observes.* Answering what model parameters can the robot learn and in how much time reduces to answering *when* and *what* states our dynamical system can reach.

In our formulation, a designer can instantiate a parameterized model of the human (e.g. human driving a car may drive straight or take a left turn at an intersection), an online learning algorithm initialization (e.g. a uniform prior over the human taking a left or driving straight in a Bayesian learning setup), and a desired level of confidence in a particular hypothesis (e.g. acquire high probability on the human taking a left turn), and finally a measurement frequency with which the robot receives data about the human (e.g. human observations are acquired at 10 Hz).

With the components from above, answering analysis questions amounts to solving an optimal control problem whose solution determines which human observations "steer" the learning system into desired states. To determine what our robot could possibly learn, we start our dynamical system with the current estimate of the parameter and solve *forward in time* for the set of hypotheses that are reachable in finite time and observations. We can also compute the minimum (or maximum) time to learn a parameter by solving an optimal control problem *backwards in time* for the earliest time at which there exists a sequence of data points that steer the learning system to the desired parameter.

We demonstrate a variety of use cases for our analysis tool, answering these questions for an autonomous car operating at an intersection near a human-driven vehicle with unknown intention, and a navigation task in a bookstore environment around a human with unknown goals or proxemics preferences. While in this paper we focus on low-dimensional parametrizations (e.g. unknown human goal), we are encouraged by this first-of-its kind tool for analyzing adaptive human models and are excited to explore approximate DP applications that can extend our tool to higher dimensions.

7.1 Related Work

Learning from human data. Learning from human data has become increasingly popular in robotics, enabling robots to effectively predict human motion like walking or reaching [238, 117, 102, 177], to learn human preferences from physical interaction [19, 33], or to estimate the fidelity of a predictive model [74, 77]. While research on analyzing algorithms and verifying models *already* learned *offline* from human data has garnered some interest [37, 185], to date there has not been research analyzing what models *could* learn.

Analysis of learning algorithms. The advent of modern machine learning has spurred a resurgence of interest in analyzing learning algorithms. While there is a long history of relating ODEs to optimization methods [159], recently, control theoretic tools such as control Lyapunov functions and reachability analysis have been used to prove convergence and safety properties of gradient-based learning methods [222, 129, 195], POMDPs [4], and neural networks [180]. While related, our work focuses not only on analyzing learning algorithms which use human data but also leverages control tools which have not yet been studied in learning contexts and have the ability to compute time-to-reach properties.

Verification & control of dynamical systems. The control theory and formal methods communities have a rich history of verifying the behavior of dynamical systems. In contrast to the above learning algorithm analyses, these approaches not only focus on convergence and safety, but also on properties such as minimum time-to-reach [226], robustness to disturbances [25], and controller synthesis [150]. This research lays the foundation for our work wherein we model learning from human data as a dynamical system and we answer analysis queries via solving optimal control problems.

7.2 Problem Formulation & Solution

7.2.1 States & Dynamics

The robot observes data in the form of state-action pairs $(x_{\mathrm{H}}, u_{\mathrm{H}})$. Let the human's discrete-time dynamics be:

$$x_{\mathrm{H}}^{t+\Delta t} = f_{\mathrm{H}}(x_{\mathrm{H}}^t, u_{\mathrm{H}}^t) \tag{7.1}$$

where $x_{\mathrm{H}} \in \mathcal{X}$ and $u_{\mathrm{H}} \in \mathcal{U}$. Additionally, Δt represents the frequency with which our robot observes data about the human (e.g. frequency of the state estimator).

Let $\theta \in \Theta$ denote the unknown human model parameter where Θ is a discrete set of values that θ can take on. This parameter could denote a variety of unknown aspects governing human behavior, from how passive or aggressive a person drives [186], what locations in a room they are moving towards [238], how optimally the person behaves [77], or even the kind of visual cues they pay attention to in a scene [117].

The robot uses an online learning algorithm to estimate the human's intent given a sequence of observations over time. Our key idea in this work is to view the robot's learning algorithm as a dynamical system where the parameter estimate is treated as state and the human's data is control input. Let $\hat{\theta}$ be the learning algorithm's estimate. This could be, for example, a point estimate of θ or the belief $b(\theta)$.

Let the discrete-time dynamics of a learning algorithm be

$$\hat{\theta}^{t+\Delta t} = f_L(\hat{\theta}^t, x_{\mathrm{H}}^t, u_{\mathrm{H}}^t) \tag{7.2}$$

where $f_L : \mathcal{E} \times \mathcal{X} \times \mathcal{U} \rightarrow \mathcal{E}$ is a single update of $\hat{\theta}$ given $(x_{\mathrm{H}}^t, u_{\mathrm{H}}^t)$ and $\hat{\theta} \in \mathcal{E}$ is the space of estimates (e.g. $\mathcal{E} = \Theta$ for a point-estimator or $\mathcal{E} = [0,1]^{|\Theta|-1}$ for the full belief).

To capture the joint evolution between the learning algorithm's estimate of the human's intent parameter and the human's possible behavior, we consider the joint state: $x := [x_{\mathrm{H}},\ \hat{\theta}]^{\top} \in \mathcal{Z} = \mathcal{X} \times \mathcal{E}$. The human data – the actions the human takes – evolve both the physical human dynamics and the learning dynamics. Thus, we would like to analyze the joint dynamical system

$$x^{t+\Delta t} = f(x^t, u_{\mathrm{H}}^t) := \begin{bmatrix} f_{\mathrm{H}}(x_{\mathrm{H}}^t, u_{\mathrm{H}}^t) \\ f_L(\hat{\theta}^t, x_{\mathrm{H}}^t, u_{\mathrm{H}}^t) \end{bmatrix}. \tag{7.3}$$

Note that this should not be confused with the robot being able to control the human's actions. This joint system representation allows us to answer reachablility questions about which estimates are "learnable" under different restrictions on the type of data the human could produce.

7.2.2 Solution Method

Equipped with our joint dynamics which describe the evolution of the human's state and the learning algorithm, we can now formulate an optimal control problem whose solution captures which human observations "steer" the learning system into desired states. We build on our framework from [27, 14] and adapt it for analyzing general discrete-time algorithms that learn online from human data.

We define the discrete-time optimal control problem and its associated value function as

$$V(x) := \min_{\mathbf{u}_{\mathrm{H}}} \min_{t \in \{0, \Delta t, \dots, T\}} l(\xi^t(x, \mathbf{u}_{\mathrm{H}})) \tag{7.4}$$

where $\mathbf{u}_{\mathrm{H}} = [u_{\mathrm{H}}^0, \dots, u_{\mathrm{H}}^{T-1}]^{\top}$ is a sequence of controls over the time horizon and $\xi^t(x, \mathbf{u}_{\mathrm{H}})$ is the joint state achieved by applying $\mathbf{u}_{\mathrm{H}}$. Here, the function $l(\cdot)$ encodes the distance between the current system state and the desired values of the parameters we seek to estimate. Different analysis questions simply become different instantiations of this function, as we describe in Sec. 7.3. Intuitively, this value function captures the *closest* our system ever gets to the target states as measured by the signed distance function l.

This minimum-payoff optimal control problem can be solved via the principle of dynamic programming [158]. Specifically, for a finite horizon $t \in \{0, \Delta t, \dots, T\}$, the discrete-time time-dependent terminal-value Hamilton-Jacobi-Bellman variational inequality [155, 75] is:

$$\begin{aligned} V^t(x) &= \min \left\{ l(x),\ \min_{u_{\mathrm{H}} \in \mathcal{U}^t} V^{t+\Delta t}(f(x, u_{\mathrm{H}})) \right\}, \\ V^T(x) &= l(x),\ \ \forall x \in \mathcal{Z} \end{aligned} \tag{7.5}$$

where $\mathcal{U}^t$ is the set of allowable actions–and therefore data–the human can generate (described in detail in Sec. 7.3). Intuitively, this value function definition can be thought of as analogous to the discrete-time Bellman equation with discount factor $\gamma = 1$, no running cost, and only terminal cost $l(\cdot)$.

Given this time-dependent value function and a candidate initial condition of the joint state x^0, we can extract two important quantities. First, the optimal control at x is:

$$u_{\mathrm{H}}^t(x) = \arg \min_{u_{\mathrm{H}} \in \mathcal{U}^t} V^{t+\Delta t}(f(x, u_{\mathrm{H}})). \tag{7.6}$$

Secondly, we define the *time-to-learn* (TTL) as the earliest time when our system will reaches the target parameters we seek to learn starting from x^0. More formally, this TTL:

$$TTL = \min\{t : V^{T-t}(x^0) \leq 0\}. \tag{7.7}$$

7.3 Encoding Analysis Questions

We can now turn our attention to mathematically encoding the analysis questions we are interested in answering. The key components that enable us to encode analysis questions in Eq. (7.5) are (1) the target set of states $\mathcal{L}$ which is encoded via $l(\cdot)$, (2) the set of human actions, $\mathcal{U}^t$, and (3) the strategy of the human (if they are minimizing or maximizing value).

Time-to-learn (TTL) queries. Computing the best and worse-case time to learn (TTL) depends not only on the learning algorithm, but also on the value of the parameter we are trying to estimate and the type of data the robot could observe. For simplicity, we first consider computing the TTL for a specific parameter value, θ^*, and later discuss how to integrate multiple TTL estimates.

We can mathematically embed the objective of learning θ^* in a target set defined in joint state space: for example, $\mathcal{L} = \{x : x_{\mathrm{H}} \in \mathcal{X}, ||\hat{\theta} - \theta^*|| \leq \epsilon\}$. Intuitively, these target sets encode that we want our parameter estimate to be close to the true value, but we do not care where the human ends up in physical state-space as long as we have estimated the parameter well. Defining $\mathcal{L}$ to be a closed set in $\mathcal{Z}$ allows us define a function $l(x) : \mathcal{Z} \to \mathbb{R}$ such that $\mathcal{L} = \{x : l(x) \leq 0\}$. For example, the signed distance function to $\mathcal{L}$ satisfies this property. This function $l(z)$ serves as the cost function for our optimal control problem from (7.4).

Next, how quickly the robot learns also depends on the possible data about the human the robot observes. Since the observed data is the human behavior, we must define the set of controls $\mathcal{U}^t$ that the human could possibly generate at each time step. In general this set can be chosen in a variety of ways. One of the most straightforward sets is to assume that the robot could observe the human taking *any* action, i.e. $\mathcal{U}^t = \mathcal{U}$. Allowing the human to take any action–and therefore produce any data–leads to very a conservative worst-case TTL, since it allows for the human to abandon any true intent and act purely adversarially. While type of analysis may be desirable in some scenarios, a potentially more realistic set of observations could be $\mathcal{U}^t = \{u_{\mathrm{H}} : P(u_{\mathrm{H}} \mid x_{\mathrm{H}}; \theta^*) \geq \delta\}$ where $P(u_{\mathrm{H}} \mid x_{\mathrm{H}}; \theta^*)$ encodes a state and intent-conditioned action distribution. By varying $\delta \in [0, 1]$, the human is allowed to generate more or less sub-optimal data with respect to the intent-driven model.

Finally, the human's strategy in the control problem determines if we obtain best or worst-case learning estimates.

Best-case TTL (min**).** Since our target set $\mathcal{L}$ encodes the true human intent parameter that the robot wishes to learn, modelling the human as *minimizing* the value which is propagated from l in Eq. 7.5 encodes a cooperative human. That is, the human is generating data in an attempt to *help* the robot learn their true intent parameter. This enables us to extract the best-case TTL_{θ^*} via Eq. (7.7). When interested in the best-case time to learn *any* $\theta^* \in \Theta$, we can obtain a conservative best-case TTL via $\max_{\theta^* \in \Theta} TTL_{\theta^*}$.

Worst-case TTL (max**).** Alternatively, by modelling the human as *maximizing* the value, we can easily encode adversarial behavior where the human is trying to prevent the robot from learning their true θ for as long as possible. This enables us to extract the worst-case TTL_{θ^*} via Eq. (7.7). Similarly to above, we obtain an upper-bound on learning any $\theta^* \in \Theta$ by computing $\max_{\theta^* \in \Theta} TTL_{\theta^*}$.

7.3.1 Reachable parameter queries.

Computing the set of forward reachable parameters given an initial estimate follows a similar setup as the TTL queries. The main difference comes from $\mathcal{L}$, which now encodes the initial joint state. For example, $\mathcal{L} = \{x : x_{\mathrm{H}} = x_{\mathrm{H}}^0, \hat{\theta} = \hat{\theta}^0\}$. Now, the sub-zero level set of V in Eq. (7.5), encodes the set of states our system can *reach* in T time starting from $\mathcal{L}$.

7.4 Use Cases of Our Analysis Tool

We now demonstrate a variety of use cases for our tool, ranging from autonomous driving to gradient-based learning.

7.4.1 Synthesizing Safe & Efficient Contingency Planners

Motion planners for autonomous vehicles often face uncertainty in how human-driven vehicles will behave. Consider the scenario where an autonomous vehicle is attempting to turn left at an unsignalized four-way intersection and there is a human-driven vehicle in the opposing lane (see Fig. 7.1). The autonomous vehicle has uncertainty about whether the human will turn left or go straight (goal 1 and goal 2, respectively)– the outcome of which significantly impacts the autonomous car's ultimate maneuver. This is therefore a prediction problem in which the model's parameter represents the human's maneuver goal: $g := \theta$ and $g \in \Theta = \{g_1, g_2\}$.

A safe solution to this problem computes a plan for the autonomous vehicle which safeguards against both events during the entire planning horizon, i.e. avoids collisions with the straight and left-turn trajectory. While safe, this approach often yields conservative and inefficient motions (left Fig. 7.1).

Alternatively, recent methods have proposed *contingency planning* [91]. In contingency planning, the robot generates a plan which safeguards against all events for a short horizon which is less than the entire planning horizon: $t_b < N$. After t_b, the motion planner generates $|\Theta|$ contingency plans, each of which safeguards against only a single event. The premise of contingency planning is that the robot will know by t_b which branch to choose, and will no longer need to safeguard against all events. The contingency plan is thus only safe if the robot gains enough certainty to choose which plan to use by the the branching time. If the branching time is too short, then when it comes time for the robot to choose a contingency plan, the robot will still have uncertainty over the human's goal. The robot could now be in a position where no plan exists which safeguards against *all* events that are still likely (center, Fig. 7.1).

In [91], the contingency planning problem is posed as a nonlinear constrained optimization problem which jointly optimizes over the shared segment and the contingency plans while weighting each contingency plan proportional to the belief in that event occurring

$$\begin{aligned} \arg\min_{\mathbf{x}_\mathrm{R}, \mathbf{u}_\mathrm{R}} \quad & J_{share}(x_\mathrm{R}^{0:t_b}) + \sum_{g \in \Theta} b(g) J_{cont}(x_\mathrm{R}^{t_b+1:N}, g) \\ \text{s.t.} \quad & x_\mathrm{R}^{t+1} = f_\mathrm{R}(x_\mathrm{R}^t, u_\mathrm{R}^t), \ \forall t \in [0, N] \end{aligned} \tag{7.8}$$

where $\mathbf{x}_\mathrm{R} \in \mathbb{R}^{n_\mathrm{R} \times N}$ denotes a vector containing the robot's planned state trajectory. Here, the robot's dynamics f_R are modelled by a 3D Dubins' car where the linear and angular velocity are control inputs and the state is the position and heading. The cost functions J_{share} and J_{cont} encode costs for colliding with static obstacles, reaching the robot's goal, and large changes in acceleration. Additionally, J_{share} penalizes collisions with *all* of the possible outcomes, while J_{cont} penalizes collisions with the human's predicted trajectory towards only the relevant goal, g_i. A critical design parameter when it comes to the safety of such a motion planning scheme is the choice of branching time, t_b.

Unfortunately, knowing the future time at which the robot will have certainty about the human's intent is in general challenging. This is where we leverage our analysis framework: we can use it to compute the *worst-case* time it will take the robot to gain certainty in the human intent.

Human Dynamics and Intent Model. Let the human-driven vehicle be modelled as a 3D Dubins' car with planar position and heading as state and discrete-time dynamics as

$$x_\mathrm{H}^{t+\Delta t} = x_\mathrm{H}^t + \Delta t \begin{bmatrix} v\, cos(\phi) \\ v\, cos(\phi) \\ u_\mathrm{H} \end{bmatrix} \tag{7.9}$$

where the human's control is angular velocity, $u_\mathrm{H} \in \{-3.5, 0, 3.5\}$ rad/s and is driving with a fixed speed $v = 6$ m/s. The human is modelled as choosing actions via the

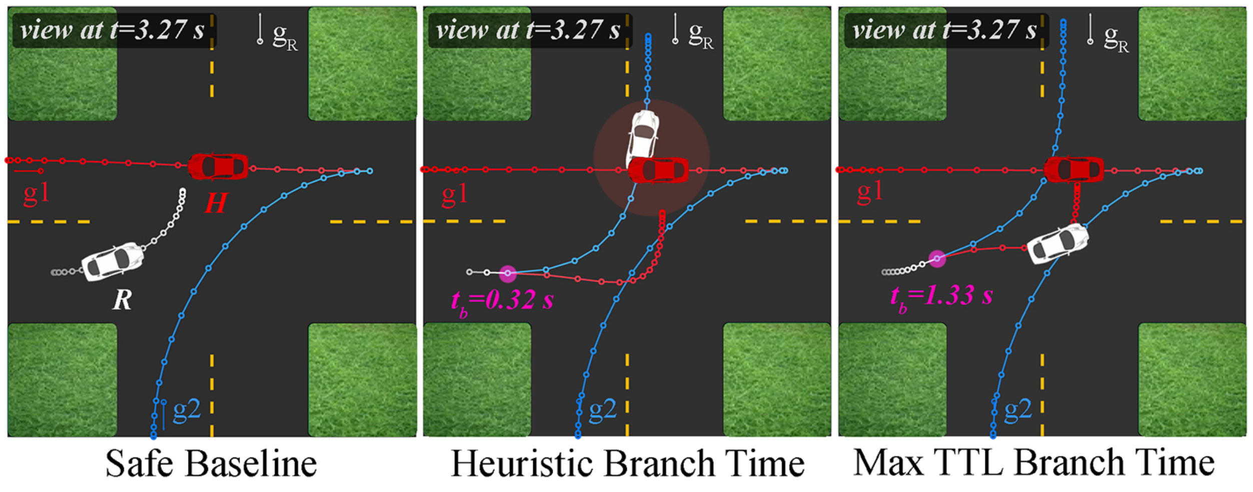


Figure 7.1: (left) Robot safeguards against both hypotheses, straight and left, for entire planning horizon, (center) A heuristically chosen branch time for the contingency planner doesn't allow the robot to observe enough human data, leading it to collision, (right) Contingency planner branches at the max TTL computed via our method, enabling a safe but efficient plan.

	Prior	Efficiency (dist to robot's goal)	Safety (min dist between cars)
Safeguard Both	n/a	6.88 m (1.14)	0.73 m (0.95)
Heuristic (0.3265 s)	Correct	5.13 m (1.89)	0.19 m (0.21)
	Incorrect	5.13 m (1.89)	-1.55 m (0.99)
	Uniform	6.10 m (0.79)	-0.76 m (1.19)
Max TTL (ours)	Correct	2.33 m (2.29)	0.58 m (0.77)
	Incorrect	5.25 m (2.88)	0.54 m (0.88)
	Uniform	3.42 m (3.13)	0.55 m (0.77)

Table 7.1: Autonomous driving experiment results shown averaged across initial conditions and ground-truth human goals. Mean efficiency and safety metrics are reported in each row and standard deviation in parenthesis.

noisily-rational model [21]: $P(u_{\mathrm{H}} \mid x_{\mathrm{H}}, g) \propto e^{Q(x_{\mathrm{H}}, u_{\mathrm{H}}; g)}$ where $Q(x_{\mathrm{H}}, u_{\mathrm{H}}; g)$ encodes the state-action value for the human's driving goal.

Robot Learning Algorithm. The robot learns the human intent by maintaining and updating a Bayesian belief over g. Since g is discrete, the robot can only maintain $|\Theta|-1 = 1$ probabilities. Without loss of generality, let the robot update $b(g = g_1) := \hat{\theta}$. The learning dynamics f_L in Eq. (7.2) are

$$f_L(b^t(g_1), x_{\mathrm{H}}^t, u_{\mathrm{H}}^t) := \frac{1}{Z} P(u_{\mathrm{H}}^t \mid x_{\mathrm{H}}^t, g_1) b^t(g_1) \tag{7.10}$$

where Z is the normalizer.

Joint State & Dynamics. The joint state is $x = [x_{\mathrm{H}}, b(g_1)]^{\top}$ and the joint dynamics are the stacked dynamics equations from above. We use $\Delta t = 0.0891s$ in all simulations.

Target Set. To determine the maximum time it will take our robot to estimate that the human is going forward (g_1) or left (g_2), we define two target sets in our joint state space:

$$\mathcal{L}_{g_1} = \{x : x_{\mathrm{H}} \in \mathcal{X}, b(g_1) \geq 0.9\} \tag{7.11}$$
$$\mathcal{L}_{g_2} = \{x : x_{\mathrm{H}} \in \mathcal{X}, 1 - b(g_1) \geq 0.9\} \tag{7.12}$$

Each of these sets encodes that the robot must be at least 90% confident in the human driving forwards or turning left.

Computing the Worst-Case Time-to-Learn (TTL). Here, we model the human as adversarial and therefore use a max in the inner value function update from Eq. (7.5). To ensure that while the person is being adversarial, they have to eventually complete their maneuver, we restrict the set of controls to $\mathcal{U}^t = \{u_{\mathrm{H}} : P(u_{\mathrm{H}} \mid x_{\mathrm{H}}, g^*) \geq 0.27\}$ where g^* is equal to the human intent which is being analyzed. Over a horizon of $T = 1.7820s$, we perform two value function computations via Eq. (7.5) and compute worst-case TTL for g_1 and g_2 by searching backwards in time for the earliest time at which the human's initial state and the robot's initial prior appears in the sub-zero level set of $V^t(x)$ (as in Eq. (7.7)) to obtain TTL_{g_1} and TTL_{g_2}. Finally, to determine the final safe branching time, we want to safeguard against the hypothesis which takes the longest to estimate confidently. Thus, let $t_b = \max\{TTL_{g_1}, TTL_{g_2}\}$.

Results. We ran a series of simulations comparing the safe motion planner, a heuristically-chosen branching time (comparable to [91]), and our maximum TTL branching time contingency planners. For all planners, we varied the initial velocity of the human and robot cars (stopped, moving slowly, or moving quickly) as well as the two possible goals the simulated human was actually moving to. For the contingency planners, we also varied the prior to correctly biased to the human's true goal, incorrectly biased, or uniform.

Table 7.1 summarizes the average efficiency and safety metrics over each of these trials. Here, the heuristically-chosen branching time branches too early – this does not allow the robot to collect enough observations about the human's behavior for the robot to make a confident but safe maneuver, which results in collisions when the robot begins with either a uniform or incorrect prior over the human goals. In contrast, the maximum worst-case TTL safeguards against both events for a longer time horizon during which the robot collects enough observations to make safe and efficiently goal-driven plans even with a uniform or incorrect prior. Note that the heuristic branching time serves to demonstrate how choosing an uninformed t_b can lead to safety violations. However, our analysis tool should be thought of as complementary to [91] wherein we can synthesize an informed t_b.

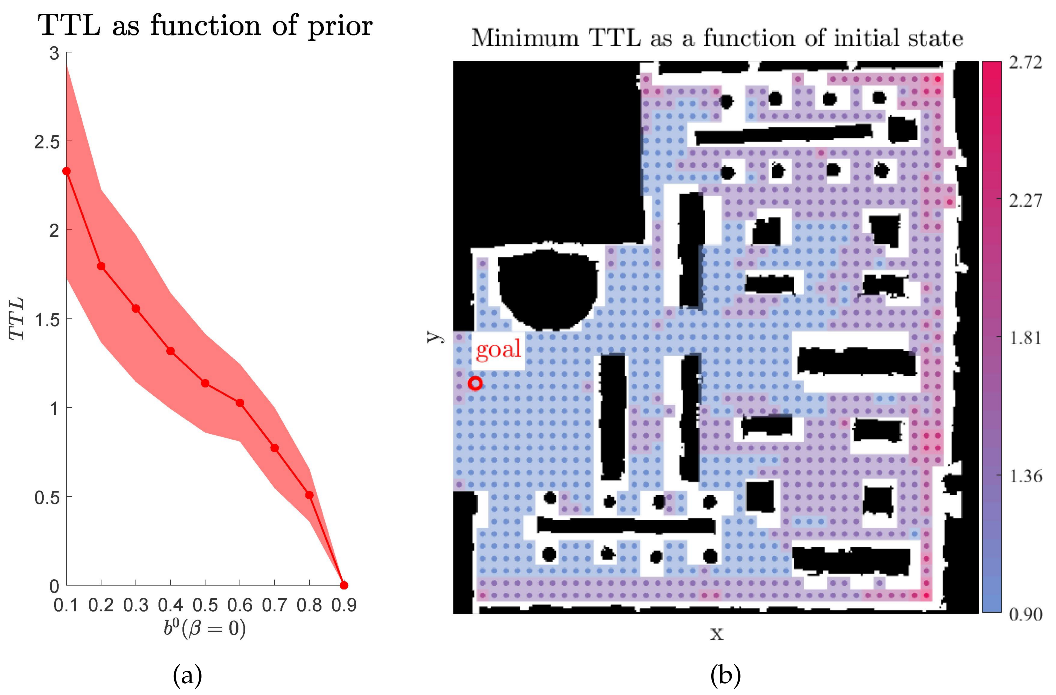


Figure 7.2: (left) Minimum TTL that the human is being unmodelled as a function of the prior. Mean and standard deviation shown in red. (right) Minimum TTL as a function of x_H with a uniform prior. Occupancy map of the bookstore environment (in Fig. 7.4) shown with modelled human goal is red circle and the min TTL for each initial (x,y) state is shown in a color ranging from blue (TTL=0.9s) to red (TTL=2.72s).

7.4.2 Analyzing Confidence-Aware Predictors

Next, we consider predictors that introspect on their model confidence, and use our tool to analyze how long it takes such a predictor to detect that its model cannot explain the observed human behavior. We focus on intent-driven predictors, which model human actions as noisily-optimal with respect to cumulative reward, $P(u_H \mid x_H; \beta) \propto e^{\beta Q(x_H, u_H)}$ [21, 237]. Here, the parameter β models how optimally the human behaves: high values of β model near-optimal behavior, whereas $\beta = 0$ removes the influence of the modelled reward on the human's behavior entirely. Recent work [74, 77, 15] proposed that rather than fixing β, the robot should estimate it. Upon observing human actions that are poorly explained by the reward function, low values of β (signaling low model confidence) will be the most likely, and our predictor will make higher-variance predictions, accounting for its inability to explain the human's behavior.

As the human deviates from the model's assumptions and the predictor makes higher variance predictions, the robot must stay further away from the human to avoid colliding with the now larger set of sufficiently likely states. This begs the question: when the human doesn't actually optimize the modeled reward, how long will it take the predictor to adapt its β and detect that its model confidence is low?

Here, let the unknown parameter be $\beta := \theta$ with $\beta \in \Theta = \{0, 1\}$, signaling low and high model confidence respectively.

Human Dynamics and Intent Model. Let the human be modelled by a simple planar pedestrian model: $x_{\mathrm{H}}^{t+\Delta t} = x_{\mathrm{H}}^{t} + \Delta t[v_H \cos(u_{\mathrm{H}}), v_H \sin(u_{\mathrm{H}}),]^\top$ where the human's control is their heading, $u_{\mathrm{H}} \in \{-\pi, \dots, \pi\}$ and the human is walking at a leisurely speed ($v = 0.6\ m/s$) or is stopped ($v = 0$). The human's reward function encourages motion towards the door (shown in red) in the indoor environment (top-down occupancy map shown on right of Fig. 7.2).

Robot Learning Algorithm. The robot maintains and updates a Bayesian belief over the confidence parameter β. Without loss of generality, let the robot explicitly update $b(\beta = 0) := \hat{\theta}$. Thus, f_L is identical to (7.10) but with $b(\beta = 0)$.

Joint State & Dynamics. The joint state is $x = [x_{\mathrm{H}}, b(\beta = 0)]^\top$ and the joint dynamics are the stacked physical and learning dynamics from above. Finally, we use $\Delta t = 0.4545\ s$.

Target Set. We are primarily interested in determining how long it will take our robot to estimate that our model cannot explain the human's behavior ($\beta = 0$). Thus, we are interested in answering "From the prior over the model confidence, what is the *fastest* our robot could learn that the person is behaving in an unmodelled way?" Our corresponding target set encoding this question is $\mathcal{L}_{\beta=0} = \{x : x_{\mathrm{H}} \in \mathcal{X}, b(\beta = 0) \geq 0.9\}$ where $\epsilon = 0.9$ is our desired confidence.

Computing the Best-Case Time-to-Learn (TTL). We seek to compute fastest time to learn we have low confidence in our model, since this is the lower bound on reaction time to unmodelled data. Thus, we use min over $\mathcal{U}^t$ in the optimization from (7.5) and we optimize over all data the human could generate, since in the worst case the human is not behaving according to the specified reward function at all. In the following two analyses, the human navigates in a bookstore environment [**aws2020environments**] whose occupancy map is shown in right of Fig. 7.2 and 3D model is shown in Fig. 7.4.

Results: Best-case TTL as function of *prior*. We first analyzed the min TTL a low model confidence as a function of the prior. After computing one backwards reachability computation, we extracted the TTL for 121 initial x_{H} states and 8 levels of the prior. The mean and standard deviation across all initial conditions shown in the left Fig. 7.2. This analysis reveals the added difficulty for the robot to detect its model is wrong if it begins with an optimistic prior.

Results: Best-case TTL as function of *initial human state*. Right of Fig. 7.2 shows how the best-case TTL varies as function of the initial x_{H} in a complex environment. Here we fix a uniform prior over the model confidence and use the same value function computed from above to query for the TTL for 1,010 initial human states. Interestingly, this analysis demonstrates that the best-case TTL is largely impacted by the constraints of the physical environment. If the human begins in the open-space near the door, learning that they do not want to move to the door is easy since the human can directly move away from the door to indicate this mismatch. However, if the human begins in a heavily constrained part of the environment such as the lower right-hand corner, all collision-free actions appear ambiguous under the robot's likelihood model; that is, moving left or up could either indicate the human intends to move to the door or they intend to move in a completely different direction.

A few interesting takeaways for designers of robot motion planners which rely on confidence-aware models include: (1) if the robot does not re-plan faster than this best-case TTL then the robot will not be able to react quickly enough to the human's unmodelled behavior and (2) the robot should remain more cautious around the human in constrained parts of the environment due to the increased learning uncertainty.

7.4.3 Generating Legible & Deceptive Behaviors

So far we demonstrated how to compute the worst and best-case learning times. We now showcase how our analysis tool can also synthesize the behavior which led to the fastest or slowest learning by our (robot) observer.

We use the same planar pedestrian model from 7.4.2. In our running example, the human is walking around a living room (occupancy map shown in Fig. 7.3) and the robot has uncertainty about which location the human is navigating to. Let the uncertain human model parameter be $g := \theta$ and $g \in \Theta = \{g_1, g_2\}$. The robot uses a noisily-rational model as above, parameterized by g, and learns via a Bayesian update.

We perform four reachability computations, two of which have a high-confidence in g_1 target set as in (7.11) and two of which have a high-confidence in g_2 target set (like (7.12)). However, for each pair of reachability computations, we perform one computation where the human is *minimizing* the value (i.e. helping robot learning) and the other where the human is *maximizing* the value (i.e. trying to slow down robot learning). In all examples, the human chooses from $\mathcal{U}^t = \{: P(|, g^*) > 0.15\}$ where g^* the goal being analyzed. The resulting four value functions are used via Eq. (7.6) to extract the optimal sequence of human data which lead to best and worst-case learning times for g_1, g_2.

Fig. 7.3 visualizes the optimal controls in the bright colored trajectory corresponding to the human's true goal. We contrast this with the optimal only-goal-driven policy in grey. Inset plots show $b(g_1)$ over time and the target confidence level plotted as a dashed line. To increase the probability on g_1 as fast as possible, the human moving to g_1 signals this by quickly moving to their left (instead of moving forward as in the optimal path). This is in line with prior work on legibility [64], but a potential advantage of our formulation is that the objective of minimizing TTL is task-oriented. While prior work encouraged agents to be legible along the entire trajectory, here the agent is directly minimizing the time to convey the goal, so that the observer can react as quickly as possible. Interestingly, to be deceptive, the human zig-zags to confuse the observer for the longest.

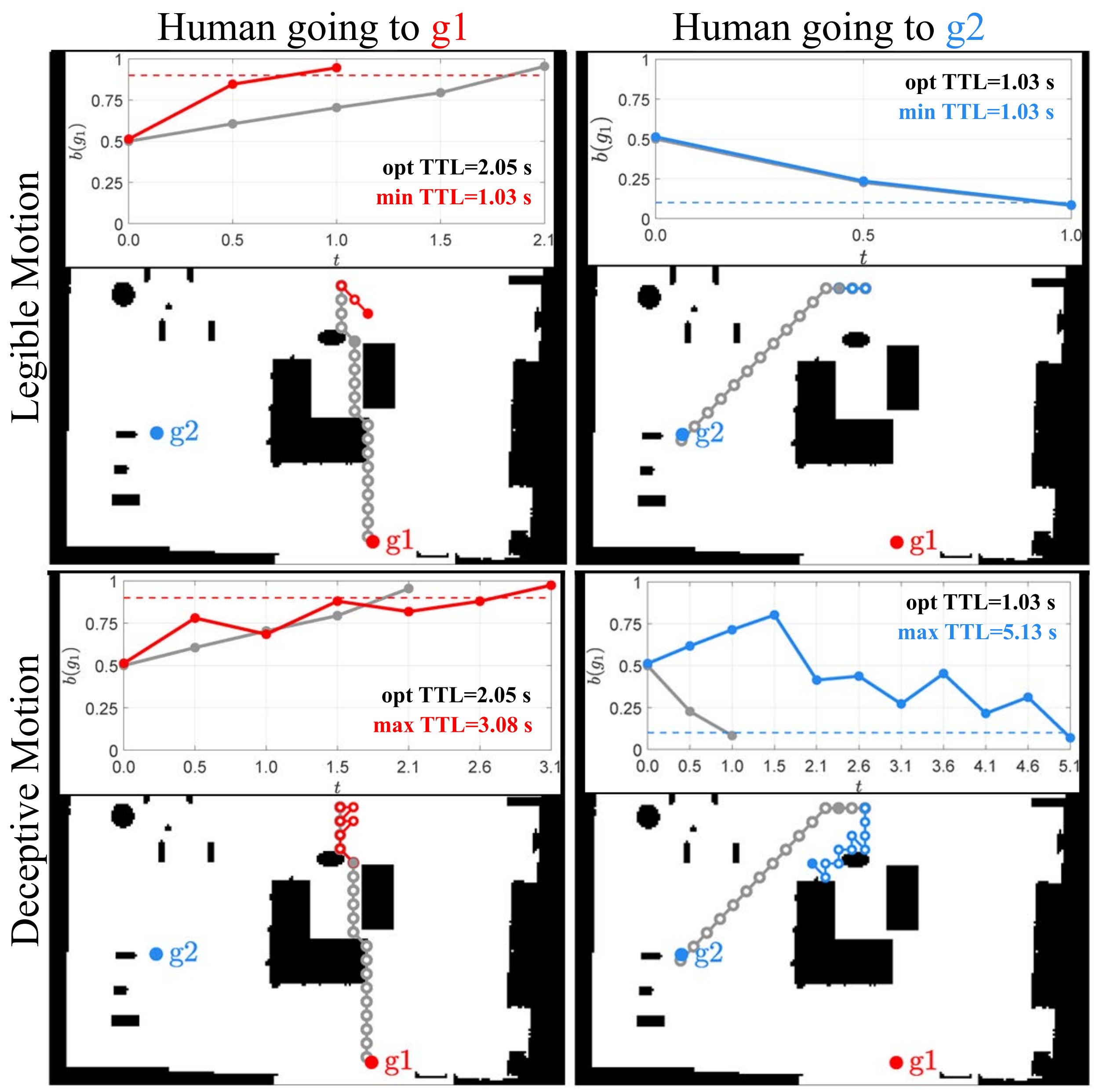


Figure 7.3: Legible and deceptive behaviors as synthesized by our analysis tool–shown in bright red or bright blue. The optimal policy for each goal is shown in grey. The estimate of the goal over time for the legible and deceptive behaviors is contrasted with the optimal policy in the inset figures.

7.4.4 Online Gradient-based Learning from People

Online gradient-based learning algorithms are also useful in many HRI domains [19, 33, 160, 229]. In this case study, we use our analysis tool to determine a parameter initialization which allows the online gradient algorithm to adapt the fastest to any true human

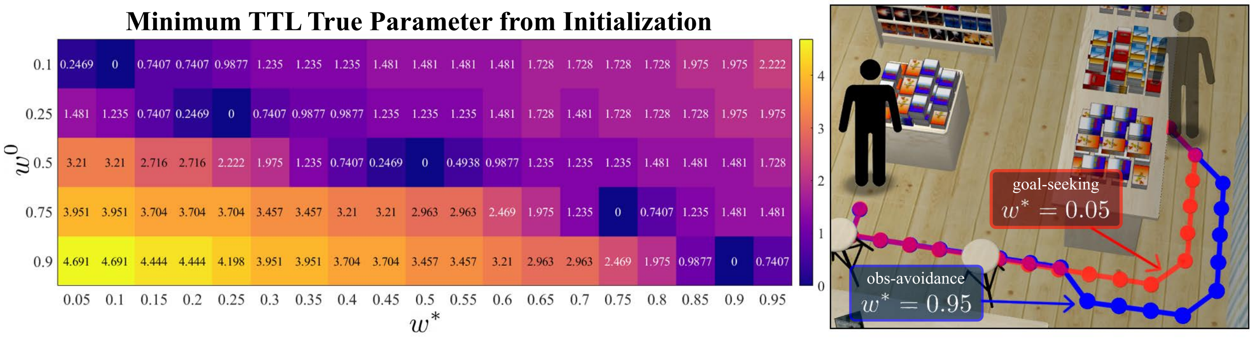


Figure 7.4: (left) Heatmap of reachable reward weights (x-axis) and their TTL (color values ranging from dark blue: $TTL = 0.0s$ to yellow: $TTL = 4.7s$) starting from a specific initialization (y-axis). (right) Bookstore environment with faded human-figure denoting the human's initial position: red path is optimal behavior for mainly goal-driven human, blue path is for a human who wants to stay far from obstacles.

intent. As above, the robot is learning a nearby human's reward function by observing their behavior. The human's reward function is modelled as a linear combination of features: $r(x_{\mathrm{H}}, u_{\mathrm{H}}; \theta) = \theta^\top \mu(x_{\mathrm{H}}, u_{\mathrm{H}})$. Here, $\theta = [1 - w,\ w]^\top$ and $\mu(x_{\mathrm{H}}, u_{\mathrm{H}}) \in \mathbb{R}^2$, encoding the distance between the human and their goal and the distance between the human and obstacles. Adjusting w trades off the human's goal-driven and obstacle-avoidance preferences.

The robot learns about the human's reward via online gradient descent. Thus, the f_L from Eq. (7.2) are:

$$f_L(\hat{\theta}^t, x_{\mathrm{H}}^t, u_{\mathrm{H}}^t) := \hat{\theta}^t + \alpha \nabla_{\hat{\theta}} F(x_{\mathrm{H}}^t, u_{\mathrm{H}}^t, \hat{\theta}^t) \tag{7.13}$$

Maximizing the likelihood of the observed $(x_{\mathrm{H}}^t, u_{\mathrm{H}}^t)$ pair under the maximum entropy distribution [237], we derive a gradient-based update:

$$F(x_{\mathrm{H}}^t, u_{\mathrm{H}}^t, \theta^t) := Q(x_{\mathrm{H}}^t, u_{\mathrm{H}}^t; \theta^t) - \mathbb{E}_{u \sim P(u \mid x_{\mathrm{H}}^t; \theta^t)} \Big[Q(x_{\mathrm{H}}^t, u; \theta^t) \Big] \tag{7.14}$$

Note: this is the state-action equivalent of learning offline from demonstrations [237].

In this analysis, we solve a *forward reachability* problem where we start our system in the target set $\mathcal{L} = \{x : x_{\mathrm{H}} = x_{\mathrm{H}}^0, \hat{\theta} = \hat{\theta}^0\}$ where x_{H}^0 is the current physical state of the human and $\hat{\theta}^0$ is a candidate initialization. We compute the set of θ's for which there exists a sequence of observations which evolve the joint system to that θ-state in finite time. Here, $\Delta t = 0.2469\ s$ and the total time horizon for which we evolve our system is $T = 7.1605\ s$.

Fig. 7.4 shows the reachable $\theta^* = [1 - w^*, w^*]$'s starting from a given $\theta^0 = [1 - w^0, w^0]$. Colors in the heatmap represent the earliest time at which the robot can learn a θ^* starting from a each initialization. Interestingly, if the robot begins with $w^0 = 0.9$ (i.e. human is primarily obstacle-averse) it takes $\sim 4.7\ s$ to learn that the person is *actually* primarily goal-seeking ($w^* = 0.1$). In contrast, initializing with $w^0 = 0.25$ allows the robot to learn *any* other parameter in $< 2.2\ s$. Intuitively, this discrepancy in how quickly the robot can learn from an initialization is because of the structure of the environment which in turn affects the gradient update. Since here the person begins in a part of the environment where their direct path is obstructed by obstacles (right Fig. 7.4), they must navigate around the obstacles before they get to the goal. The evidence of the person moving *away* from obstacles makes it difficult to disambiguate if they truly are goal-driven or obstacle-averse. Thus, initializations which bias our estimator towards believing that people are obstacle-averse is a poor choice if we want our robot to learn any other θ^* quickly.

Closing Remarks. In this work, we leveraged tools from reachability analysis to analyze human models which adapt online. By treating these models as dynamical systems where the estimate is state and the human data is control, we obtain the best and worst-case time to learn, extract the optimal measurements which enable learning, and synthesize good model parameter initializations.

Part III

Safety for HRI Beyond Collision-Avoidance

Thus far, this thesis has focused on traditional collision-avoidance notions of safety. However, close collaboration with people demands more than just collision-avoidance, raising the question: *what is the right notion of safety?* This need is highlighted in the robot learning from physical human interactions domain: after consistently misinterpreting human feedback during a household cleaning task, the robot erroneously learns to move coffee mugs at an angle, resulting in spilled coffee and miscoordination. Part III first lays the groundwork for how robots can learn online from intentional physical human corrections. We then introduce a variant of confidence-aware robot learning for the physical HRI domain, where the robot explicitly reasons about how well it can explain human corrections (or demonstrations) give it's hypothesis space. We demonstrate how this reduces incorrect robot learning when the robot's hypothesis space is not aligned with the human's during reward learning. This section is an exciting first step towards the large problem domain of formalizing safety in HRI beyond collision-avoidance.

Chapter 8

Learning Robot Objectives from Physical Human-Robot Interaction

This chapter is based on the paper "Physical Interaction as Communication: Learning Robot Objectives Online from Human Corrections" [148] written in collaboration with Dylan Losey, Marcia O'Malley, and Anca Dragan.

Physical interaction is a natural means for collaboration and communication between humans and robots. From compliant designs to reliable prediction algorithms, recent advances in robotics have enabled humans and robots to work in close physical proximity. Despite this progress, seamless physical interaction—where robots are as responsive, intelligent, and fluid as their human counterparts—remains an open problem.

One key challenge is determining how robots should *respond* to direct physical contact. Fast and safe responses to external forces are generally necessary, and have been studied extensively within the field of physical human-robot interaction (pHRI). A traditional controls approach is to treat the human's interaction force as a perturbation to be rejected or ignored. Here the robot assumes that it is an expert agent and follows its own predefined trajectory regardless of the human's actions [59]. Alternatively, the robot can treat the human as the expert, so that the human guides the passive robot throughout their preferred trajectory. Whenever the robot detects an interaction it stops moving and becomes transparent, enabling the human to easily adjust the robot's state [101]. Impedance control—the most prevalent paradigm for pHRI [87, 97]—combines aspects of the previous two control strategies. Here the robot tracks a predefined trajectory, but when the human interacts the robot complies with the human's applied force. Under this approach the human can intuitively alter the robot's state while also receiving force feedback from the robot.

In each of these different response strategies for pHRI the robot returns to its preplanned trajectory as soon as the human stops interacting. In other words, the robot remains confident that its original trajectory is the correct way to complete the task. Since this robot trajectory is optimal with respect to some underlying objective function, these response paradigms effectively maintain a fixed objective function during pHRI. Hence, the human's interactions do not change the robot's understanding of the task; instead, external forces are simply *disturbances* which should be reacted to, rather than information which should be reasoned about.

In this work we assert that physical human interactions are often *intentional*, and occur because the robot is doing something that the human believes is incorrect. The fact that the human is physically intervening to fix the robot's behavior implies that the robot's trajectory—and therefore the underlying objective function used to produce this trajectory—is wrong. Under our framework we consider the forces that the human applies as observations about the true objective function that the robot should be optimizing, which is known to the human but not by the robot. Accordingly, human interactions should no longer be thought of as only disturbances that perturb the robot from its preplanned trajectory, but rather as *corrections* that teach the robot about the desired behavior during the task.

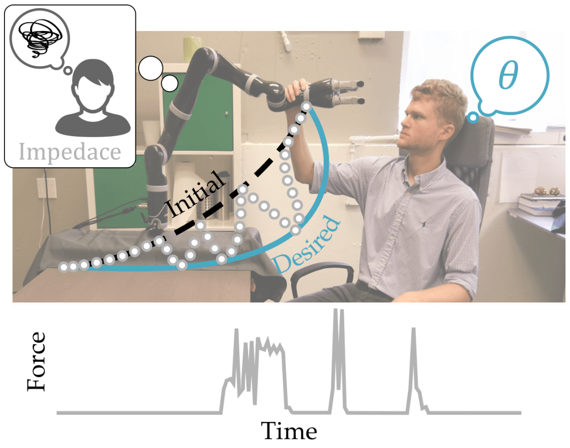
This insight enables us to formalize the robot's response to pHRI as an instance of a partially observable dynamical system, where the robot is unsure of its true objective function, and human interactions provide information about that objective. Solving this system defines the *optimal way* for the robot to respond to pHRI. We derive an approximation of the solution to this system that works in real-time for continuous state and action spaces, enabling robot arms to react to pHRI online and adjust how they complete the current task. Due to the necessity of fast and reactive schemes, we also derive an online gradient-descent solution that adapts inverse reinforcement learning approaches to the pHRI domain. We find that this solution works well in some settings, while in others user corrections are noisy and result in unintended learning. We alleviate this problem by introducing a restriction to our update rule focused on extracting only what the person intends to correct, rather than assuming that every aspect of their correction is intentional. Finally, we compare our approximations to a full solution, and experimentally test our proposed learning method in user studies with a robotic manipulator.

We make the following contributions[1]:

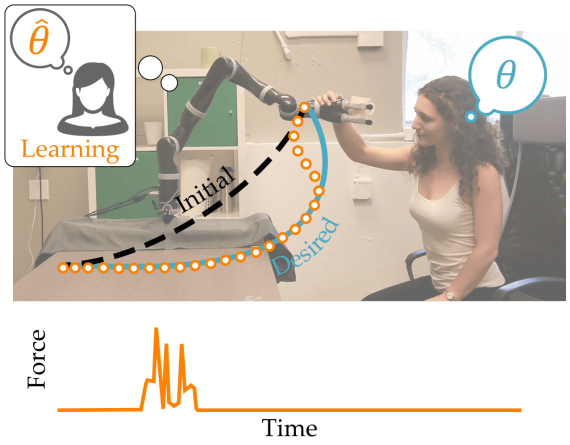
Formalizing pHRI as implicitly communicating objectives. We formalize reacting to physical human-robot interaction as a *dynamical system*, where the robot optimizes an objective function with an unknown parameter θ, and human interventions serve as observations about the true value of θ. As posed, this problem is an instance of a Partially Observable Markov Decision Process (POMDP).

Learning online from pHRI and safely controlling the robot. Responding to pHRI requires learning about the objective in real-time (the estimation problem), as well as adapting the robot's motion in real-time (the control problem). We derive an approximation that enables both by moving from the *action or policy* level to the *trajectory* level, bypassing the need for dynamic programming or POMDP solvers, and instead relying on

[1]Note that parts of this work have been published at the Conference on Robotic Learning [19] and the Conference on Human-Robot Interaction [18].



(a) Robot that treats physical interactions as disturbances.



(b) Robot that treats physical interactions as intentional and informative.

Figure 8.1: (Top) When physical human interactions are treated as disturbances people have to repeatedly push the robot to physically change its behavior. (Bottom) Robots that recognize that physical interactions may be corrections can learn from these interactions and change their underlying behavior to align with the human's preferences.

local optimization. Working at the trajectory level we derive an online gradient descent learning rule which updates the robot's estimate of the true objective θ as a function of the human's interaction force.

Responding to unintended human corrections. In practice, the human's physical interactions are noisy and imperfect, particularly when trying to correct high degree-of-freedom (DoF) robotic arms. Because these corrections do not isolate exactly what the human is trying to change, responding to all aspects of pHRI can result in *unintended learning*. We therefore introduce a restriction to our online learning rule that only updates the robot's estimate over aspects of the task that the person was most likely trying to correct.

Analyzing approximate solutions. In a series of controlled human-robot simulations we compare the performance of our online learning algorithm to the gold standard: computing an optimal *offline* solution to the pHRI formalism. We also consider two baselines: *deforming* the robot's original trajectory in the direction of human forces, and reacting to human forces with only *impedance control*. We find that our online learning method outperforms the deformation and impedance control baselines, and that the difference in performance between our online learning method and the more complete offline solution is negligible.

Conducting user studies on a 7-DoF robot. We conduct two user studies with the JACO2 (Kinova) robotic arm to assess how online learning from physical interactions affects the robot's objective performance and the user's subjective feedback. During these studies the robot begins with an incorrect objective function and participants must physically intervene mid-task to teach the robot to execute the remainder of the task correctly. In our first study we find that participants are able to physically teach the to perform the task correctly, and that participants prefer robots that learn from pHRI. In our second study we test how learning from all aspects of the human's interaction compares to our restriction, where the robot only learns about the single feature most correlated with the human's correction.

Overall, this work demonstrates how we can leverage the implicit communication which is present during physical interactions. Learning from implicit human communication applies not only to pHRI, but conceivably also to other kinds of actions that people take.

8.1 Prior Work

In this work, we enable robots to leverage physical interaction with a human during task execution to learn a human's objective function. We also account for imperfections in the way that people physically interact to correct robot behavior. Prior work has separately addressed (a) control strategies for reacting to pHRI *without* learning the human's objective and (b) learning the human's objective *offline* from kinesthetic demonstrations. An exception is work on shared autonomy, which learns the human's objective in real-time, but only when that objective is parameterized by the human's goal position. Finally, we discuss related work on algorithmic teaching, which describes how humans can optimally teach robots as well as how humans practically teach robots.

Controllers for pHRI. Recent review articles on control for physical human-robot interaction [87, 59] group these controllers into three categories: impedance control, reactive strategies, and shared control. When selecting a controller for pHRI, ensuring the human's safety is crucial. Impedance control, as originally proposed by [97], achieves human safety by making robots compliant during interactions; for instance, the robot behaves like a spring-damper centered at the desired trajectory. But the robot can react to human contacts in other ways besides—or in addition to—rendering a desired impedance. [88] suggest a variety of alternatives: the robot could stop moving, switch to a low-impedance mode, move in the direction of the human's applied force, or re-time its desired trajectory.

More relevant here are works on shared control, where the robot has an objective function, and uses that objective function to select optimal control feedback during pHRI [101, 156, 147]. In [140] the authors formulate pHRI with game theory. The robot has an objective function which depends on the error from a pre-defined trajectory, the human's effort, and the robot's effort. During the task the robot learns the relative weights of these terms from human interactions, resulting in a shared controller that becomes less stiff when the human exerts more force. Rather than only learning the correct robot stiffness—as in [140]—our work more generally learns the correct robot behavior. We note that each of these control methods [97, 88, 101, 156, 140, 147] enables the robot to safely respond to human interactions in real-time. However, once the human stops interacting, the robot resumes performing its task *in the same way* as it had planned before human interactions.

Learning Human Objectives Offline. Inverse reinforcement learning (IRL), also known as inverse optimal control, explicitly learns the human's objective function from demonstrations [3, 109, 162, 167]. IRL is an instance of supervised learning where the human shows the robot the correct way to perform the task, and the robot infers the human's objective *offline* from one or more demonstrations. Demonstrations can be provided through pHRI, where the human kinesthetically guides the passive robot along their desired trajectory [70, 108]. In practice, the human's actual demonstrations may not be optimal with respect to their objective, and [173, 237] address IRL from approximately optimal or noisy demonstrations.

Most relevant to our research are IRL approaches that learn from corrections to the robot's trajectory rather than complete demonstrations [99, 113, 175]. Within these works, the human corrects some aspect of the demonstrated trajectory during the current iteration, and the robot improves its trajectory *the next time* it performs the task. By contrast, we use human interactions to update the robot's behavior during the *current* task. Our solution for real-time learning is analogous to online Maximum Margin Planning [175] or coactive learning [99, 197], but we derive this solution as an approximately optimal response to pHRI. Moreover, we also show how this learning method can be adjusted to accommodate *unintentional* human corrections.

As we move towards online learning, we also point out research where the robot learns a discrete set of candidate reward functions offline, and then changes between these options based on the human's real-time physical corrections [228]. We view this work as a simplified instance of our approach, where the robot has sufficient domain knowledge to limit the continuous space of rewards to a few discrete choices.

Learning Human Goals Online. Prior work on shared autonomy has explored how robots can learn the human's objective *online* from the human's actions. [65, 102] consider human-robot collaboration and teleoperation applications, in which the robot observes the human's inputs, and then infers the human's desired *goal position* during the current task. Other works on shared autonomy have extended this framework to learn the human's adaptability [165] or trust [46] so that the robot can reason about how its actions may alter the human's goal. In all of these prior works the robot is moving through free-space and the human's preferred goal is the only aspect of the true objective which is unknown. We build on this prior work by considering *general objective parameters*; this requires a more complex—i.e., non-analytic and difficult to compute—observation model, along with additional approximations to achieve online performance.

Although not part of shared autonomy, we also point out research where the robot's trajectory changes online due to physical human interactions. In some works—such as [153, 201]—the robot alters its trajectory to *avoid* physical human interaction. More related to our approach are works where the robot *embraces* physical corrections to adapt its behavior. For example, in [146, 116, 115, 145] the robot maintains a parameterized desired trajectory or dynamical system, and updates the parameters in real-time to minimize the error between the resultant trajectory and the human's corrections. These works directly update the robot's desired trajectory based on corrections; by contrast, we learn a *reward function* from human corrections, which can—in turn—be used to generate dynamical systems or desired trajectories. Learning a reward function is advantageous here because it enables the robot to generalize what it has learned within the task, e.g., because the human has corrected the robot closer to one table, the robot will move closer to a second table as well.

Humans Teaching Robots. Recent works on algorithmic teaching, also referred to as machine teaching, can be used to find the optimal way to teach a learning agent [22, 83, 236]. Within our setting the human teaches the robot their objective function via corrections, but actual end-users are imperfect teachers. Algorithmic teaching addresses this issue by improving the human's demonstrations for IRL [39]. Here the robot learner provides advice to the human teacher, guiding them into making better corrections. By contrast, we focus on developing learning algorithms that match how *everyday end-users* approach the task of teaching [207, 208, 106]. Put another way, we do not want to optimize the human's corrections, but rather develop learning algorithms that account for imperfect teachers. Most relevant is [6], which shows how humans can kinesthetically correct the robot's waypoints offline to better match their desired trajectory. We similarly investigate interfaces that make it easier for people to teach robots, but in the context of applying physical forces to correct an existing robot trajectory.

8.2 Formalizing Physical Human-Robot Interaction

Consider a robot performing a task autonomously and in close proximity to a human end-user. The human observes this robot and can *physically* interact with the robot to alter its behavior. Returning to our running example from Fig. 8.1, imagine a robotic manipulator that is carrying a coffee mug from the top of a cabinet down to a table while the human sits nearby. Importantly, the robot is either not doing this task correctly (e.g., the robot is carrying the cup at such an angle that coffee will spill) or the robot is not doing the task according to the human's personal preferences (e.g., the robot is carrying the coffee too far above the table). In both of these cases the human is incentivized to physically interact with the robot and correct its behavior: but how should the robot respond? Here we formalize pHRI as a dynamical system where the robot does not know the correct objective function that the human wants it to optimize and the human's interactions are informative about this objective. Importantly, this formalism defines what it means for a robot to respond in the right or optimal way to physical human interactions. Furthermore, certain strategies for responding to pHRI can be justified as approximate solutions to this formalism.

Notation. Let x be the robot's state, u_r be the robot's action, and u_h be the human's action.

Returning to our motivating example, $x \in \mathbb{R}^n$ encodes the manipulator's joint positions and velocities, $u_r \in \mathbb{R}^m$ are the robot's commanded joint torques, and $u_h \in \mathbb{R}^m$ are the joint torques resulting from the wrench applied by the human. The robot transitions to the next state based on its deterministic dynamics $\dot{x} = f(x, u_r + u_h)$. Notice that both the robot's and human's action influence the robot's motion. In what follows we will work in discrete time, where a superscript t denotes the current timestep. For instance, x^t is the state at time t.

Objective. We model the human as having a particular reward function in mind that represents how they would like the current task to be performed. We write this reward function as a linear combination of task-related features [3, 237]:

$$r(x, u_r, u_h; \theta) = \theta \cdot \phi(x, u_r, u_h) - \lambda \|u_h\|^2. \tag{8.1}$$

In the above, $\phi \in [0,1]^N$ is a normalized vector of N features, λ is a positive constant, and $\theta \in \mathbb{R}^N$ is a parameter vector that determines the relative weight of each feature. Here θ encapsulates the true *objective*: if an agent knows exactly how to weight all the aspects of the task, then it can compute how to perform the task optimally. The first term in Equation (8.1) is the task-related reward, while the second term penalizes human effort. Intuitively, the human wants the robot to complete the task according to their objective θ—e.g., prioritizing keeping the coffee upright, or moving closer to the table—without any human intervention[2].

With this formalism the robot should take actions u_r to maximize the reward in Equation (8.1) across every timestep. This is challenging, however, because the robot *does not know* the true objective parameters θ: only the human knows θ. Different end-users have different objectives, which can change from task-to-task and even day-to-day. We thus think of θ as a hidden part of the state known only by the human. If the robot did know θ, then pHRI would reduce to an instance of a Markov decision process (MDP), where the states are x, the actions are u_r, the reward is (8.1), and the robot understands what it means to complete its task optimally. But since the actual robot is uncertain about θ, we must reason over this uncertainty during pHRI.

POMDP. We formalize pHRI as an instance of a partially observable Markov decision process (POMDP) where the true objective θ is a hidden part of the state, and the robot receives observations about θ through the human actions u_h. Formally, a POMDP is a tuple $\langle S, U, Z, T, O, r, \gamma \rangle$ where:

- S is the set of states, where $s = (x, \theta)$, so that the system state contains the robot state x and parameter θ
- U is the set of the robot actions u_r

[2]We recognize that $\|u_h\|^2$ could also be thought of as a feature in ϕ with weight λ; however, we have explicitly listed this term to emphasize that the robot should not rely on human guidance.

- Z is the set of observations (i.e. human actions u_h)
- $T(s^t, u_r^t + u_h^t, s^{t+1})$ is the transition distribution determined by the robot's dynamics (θ is constant)
- $O(s^{t+1}, u_r^t, z^{t+1})$ is the observation distribution
- $r(s^t, u_r^t, u_h^t)$ is the reward function from (8.1)
- γ is the discount factor

In the above POMDP the robot cannot directly observe the system state s, and instead maintains a belief over s, where $b(s)$ is the probability of the system being in state s. Within our pHRI setting we assume that the robot knows its state x (e.g., position and velocity), so that the belief over s reduces to $b(\theta)$, the robot's belief over θ. The robot does not know the human's true objective parameter θ, but updates its belief over θ by observing the human's physical interactions u_h.

Solving this POMDP yields the robot's optimal response to pHRI during the task[3]. We point out that this POMDP is atypical, however, because the observations u_h additionally affect the robot's reward r, similar to [102], and alter the robot's state x via the transition distribution T. Because the human's actions can change both the state and reward, solving this POMDP suggests that the robot should anticipate future human actions, and choose control inputs u_r that account for the predicted human inputs u_h, similar to [96].

Observation Model. Assuming that human interactions are *meaningful*, the robot should leverage the human's actions u_h to update its belief over θ. In order to associate the human interactions u_h with the objective parameter θ, the robot uses an observation model: $P(u_h \mid x, u_r; \theta)$. If we were to treat the human's actions as random disturbances, then we would select a uniform probability distribution for $P(u_h \mid x, u_r; \theta)$. By contrast, here we model the human as intentionally interacting to correct the robot's behavior; more specifically, let us model the human as correcting the robot to approximately maximize their reward. We assume the human selects an action u_h that, when combined with the robot's action u_r, leads to a high Q-value (state-action value) *assuming the robot will behave optimally after the current timestep*, i.e., assuming that the robot learns the true θ:

$$P(u_h^t \mid x^t, u_r^t; \theta) = \frac{e^{Q(x^t, u_r^t + u_h^t; \theta)}}{\int e^{Q(x^t, u_r^t + \tilde{u}_h; \theta)} d\tilde{u}_h} \tag{8.2}$$

Our choice of Equation (8.2) stems from maximum entropy assumptions [237], as well as the Bolzmann distributions used in cognitive science models of human behavior [21].

[3]The most general formulation for pHRI is that of a cooperative inverse reinforcement learning (CIRL) game [89], which, when solved, yields the optimal human and robot policies.

8.3 Approximate Solutions for Online Learning

Although we have demonstrated that pHRI is an instance of a POMDP, solving POMDPs exactly is at best computationally expensive and at worst intractable [107]. POMDP solvers have made significant progress [198, 203]; however, it still remains difficult to compute online solutions for continuous state, action, and observation spaces. For instance, when evaluated on a toy problem ($S = \mathbb{R}^4$, $O = \mathbb{R}^8$), recent developments do not obtain exact solutions within one second [206]. The lack of efficient POMDP solvers for large, continuous state, action, and observation spaces is particularly challenging here since (a) the dimension of our state space S is twice the number of robot DoF, $2n$, plus the number of task-related features, N, and (b) we are interested in real-time solutions that enable the robot to learn and act while the human is interacting (i.e. we need millisecond-to-second solutions). Accordingly, in this section we introduce three approximations to our pHRI formalism that enable online solutions. First, we separate finding the optimal robot policy from estimating the human's objective. Next, we simplify the observation model and use a maximum *a posteriori* (MAP) estimate of θ as opposed to the full belief over θ. Finally, when finding the optimal robot policy and estimating θ, we move from *policies to trajectories*. These approximations show how our solution is derived from the complete POMDP formalism outlined in the last section, but now enable the robot to learn and react in real-time with continuous state, action, and belief spaces.

QMDP. We first assume that θ will become fully observable to the robot at the next timestep. Given this assumption, our POMDP reduces to a QMDP [142]; QMDPs have been used by [102] to approximate a POMDP with uncertainty over the human's goal. The QMDP separates into two distinct subproblems: (a) *finding the robot's optimal policy* given the current belief $b(\theta)$ over the human's objective:

$$Q(x, u_r, b) = \int b(\theta) Q(x, u_r, \theta) d\theta \tag{8.3}$$

where $u_r^* = \arg\max_{u_r} Q(x, u_r, b)$ evaluated at every state yields the optimal policy, and (b) *updating the belief* $b(\theta)$ *over the human's objective* θ given a new observation:

$$b^{t+1}(\theta) = \frac{P(u_h^t \mid x^t, u_r^t; \theta) b^t(\theta)}{\int P(u_h^t \mid x^t, u_r^t; \tilde{\theta}) b^t(\tilde{\theta}) d\tilde{\theta}} \tag{8.4}$$

where $P(u_h^t \mid x^t, u_r^t; \theta)$ is the observation model in Equation (8.2), and $b^t(\theta) = P(\theta \mid x^{0:t}, u_r^{0:t}, u_h^{0:t})$ for $t \in \{0, 1, \dots\}$.

Intuitively, under this QMDP the robot is always exploiting the information it currently has, and never actively tries to explore for new information. A robot using the policy from Equation (8.3) does not anticipate any human actions u_h, and so the robot solves for its optimal policy as if it were completing the task in isolation. Recall that we previously pointed out that physical human interactions can influence the robot's state. In practice, however, we do not necessarily want to account for these actions when planning—the robot should not rely on the human to move the robot. Due to our QMPD approximation the robot never relies on the human for guidance: but when the human does interact, the robot leverages u_h to learn about θ in Equation (8.4). In summary, the robot only considers u_h for its information value.

MAP of θ. Ideally, the robot would maintain a full belief $b(\theta)$ over θ. Since the human's objective $\theta \in \mathbb{R}^N$ is continuous, potentially high-dimensional, and our observation model is non-Gaussian, we approximate b with the maximum *a posteriori* estimate. We will let $\hat{\theta}$ be the robot's MAP estimate of θ.

Planning and Control. Indeed, even if we had $b(\theta)$, solving (8.3) in continuous state, action, and belief spaces is still intractable for real-time implementations. Let us focus on the challenge of finding the robot's optimal policy given the current MAP estimate $\hat{\theta}$. We move from computing policies to planning trajectories, so that—rather than evaluating (8.3) at every timestep—we *plan* an optimal trajectory from start to goal, and then track that trajectory using a safe *controller*.

At every timestep t, we first *replan* a trajectory $\xi = x^{0:T} \in \Xi$ which optimizes the task-related reward from Equation (8.1) over the T-step planning horizon. If our features ϕ only depend on the state x, then the cumulative task-related reward becomes:

$$R(\xi;\theta) = \theta \cdot \Phi(\xi) = \sum_{x^t \in \xi} \theta \cdot \phi(x^t) \tag{8.5}$$

Here $\Phi(\xi)$ is the total feature count along trajectory ξ. Using the cumulative reward function in Equation (8.5), the robot finds the optimal trajectory ξ_r^t from its current estimate $\hat{\theta}^t$:

$$\xi_r^t = \arg\max_{\xi \in \Xi} \hat{\theta}^t \cdot \Phi(\xi) \tag{8.6}$$

We can solve Equation (8.6) for the optimal trajectory using trajectory optimization tools [192, 111]. Whenever $\hat{\theta}$ is updated from pHRI during task execution, the robot's trajectory will be replanned using that new estimate to match the the learned objective.

To track the robot's planned trajectory we leverage *impedance control*. Impedance control—as originally proposed by [97]—is the most popular controller for pHRI [87], and ensures that the robot responds compliantly to human corrections [58]. Let $x^t = (q^t, \dot{q}^t)$, where q^t is the robot's current configuration, and $q_r^t \in \xi_r^t$ is the desired configuration at timestep t. After feedback linearization [204], the equation of motion of a robot arm under impedance control becomes:

$$M_r(\ddot{q}^t - \ddot{q}_r^t) + B_r(\dot{q}^t - \dot{q}_r^t) + K_r(q^t - q_r^t) = u_h^t \tag{8.7}$$

Here M_r, B_r, and K_r are the desired inertia, damping, and stiffness rendered by the robot. These parameters determine what impedance the human perceives: for instance, lower K_r

makes the robot appear more compliant. In our experiments, we implement a simplified impedance controller without feedback linearization:

$$u_r^t = B_r(\dot{q}_r^t - \dot{q}^t) + K_r(q_r^t - q^t) \tag{8.8}$$

This control input drives the robot towards its desired state $x^t \in \xi_r^t$, and evaluating Equation (8.8) over all states yields the robot's policy. To summarize, we first solve the trajectory optimization problem from Equation (8.6) to get the current robot trajectory ξ_r^t, and then compliantly track that trajectory using Equation (8.8). Notice that if the robot never updates $\hat{\theta}$ then $\xi_r^t = \xi_r^{t-1}$, and this approach reduces to using impedance control to track an unchanging robot trajectory.

Intended Trajectories. Next we address the second QMDP subproblem: updating the MAP estimate $\hat{\theta}$ after each new observation. First we must find an observation model which we can compute in real-time. Similar to solving for our optimal policy with Equation (8.3), evaluating our observation model from Equation (8.2) for a given θ is challenging because it requires that we determine the Q-value associated with that θ. Previously we avoided this issue by moving from policies to trajectories. We will utilize the same simplification here to find a feasible observation model based on the human's intended trajectory.

Instead of attempting to directly relate u_h to θ, as in our original observation model, we propose an intermediate step: interpret each human action u_h via an *intended trajectory*, ξ_h, which the human would prefer for the robot to execute. We leverage trajectory deformations [146] to get the intended trajectory ξ_h from the robots planned trajectory ξ_r and the humans physical interaction u_h. Following [146], we propagate the human's interaction force along the robot's trajectory:

$$\xi_h = \xi_r + \mu A^{-1} U_h \tag{8.9}$$

where $\mu > 0$ scales the magnitude of the deformation. The symmetric positive definite matrix A defines a norm on the Hilbert space of trajectories and dictates the shape of the deformation [66]. The input vector is $U_h = u_h$ at the current time, and $U_h = 0$ at all other times. During experiments we use the velocity norm for A [66], but other options are possible.

Our deformed trajectory minimizes the distance from the previous trajectory while keeping the end-points the same and moving the corrected point to its new configuration [66]. Whereas using the Euclidean norm to measure distance would return the same trajectory as before with the current waypoint teleported to where the user corrected it, using a band-diagonal norm A (e.g., the velocity norm) serves to couple each waypoint along the trajectory to the one before it and the one after it. This formalizes the effect proposed by elastic strips by [36] and elastic bands by [170].

Now rather than evaluating the Q-value of $u_h + u_r$ given θ, like we did in Equation (8.2), we can compare the human's intended trajectory ξ_h to the robot's original trajectory ξ_r and relate these differences to θ. We assume that the human provides a intended trajectory ξ_h that approximately maximizes their cumulative task-related reward from Equation (8.5) while remaining close to ξ_r:

$$P(\xi_h \mid \xi_r; \theta) \approx \frac{e^{R(\xi_h;\theta)-\lambda\|\xi_h-\xi_r\|^2}}{\int e^{R(\tilde{\xi}_h;\theta)-\lambda\|\tilde{\xi}_h-\xi_r\|^2} d\tilde{\xi}_h} \tag{8.10}$$

Moving forward we treat $P(\xi_h \mid \xi_r; \theta)$ as our observation model. Note that this observation model is analogous to Equation (8.2) but in trajectory space. In other words, Equation (8.10) yields a distribution over intended trajectories given θ and the current robot trajectory. Here the correspondence between the human's effort $\|u_h\|^2$ and the change in trajectories $\|\xi_h - \xi_r\|^2$ stems from the deformation in Equation (8.9). In conclusion, we can leverage our simplified observation model (8.10) to tractably reason about the meaning behind the human's physical interaction.

8.4 All-at-Once Online Learning

So far we have determined how to choose the robot's actions given $\hat{\theta}$, the current MAP estimate of the human's objective. We have also derived a tractable observation model. Next, we apply this observation model to update $\hat{\theta}$ based on human interactions. By using online gradient descent we arrive at an update rule for $\hat{\theta}$ which adjusts the weights of all the features based on a single human correction. We refer to this method as *all-at-once* learning. We also relate all-at-once learning to prior works on online Maximium Margin Planning (MMP) and Coactive Learning.

Gradient Descent. If we assume that the observations are conditionally independent[4], then the maximum *a posteriori* (MAP) estimate at timestep $t+1$ is:

$$\begin{aligned} \hat{\theta}^{t+1} &= \arg\max_{\theta} P(\xi_h^0, \dots, \xi_h^t \mid \xi_r^0, \dots, \xi_r^t, \theta) P(\theta) \\ &= \arg\max_{\theta} \sum_{\tau=0}^{t} \ln P(\xi_h^\tau \mid \xi_r^\tau, \theta) + \ln P(\theta) \end{aligned} \tag{8.11}$$

where $P(\xi_h^\tau \mid \xi_r^\tau; \theta)$ is our observation model from Equation (8.10). To use this model we need to compute the normalizer, which requires integrating over the space of all possible human-preferred trajectories. We instead leverage Laplace's method to approximate the normalizer. Taking a second-order Taylor series expansion of $R(\xi_h, \theta) - \lambda\|\xi_h - \xi_r\|^2$ about ξ_r, the robot's estimate of the optimal trajectory, we obtain a Gaussian integral that we can

[4]Recent work by [139] extends our approach to cases where the interactions are not conditionally independent, i.e., multiple corrections are interconnected.

evaluate:

$$P(\xi_h \mid \xi_r, \theta) \approx \left(e^{R(\xi_h,\theta)-R(\xi_r,\theta)-\lambda||\xi_h-\xi_r||^2}\right) \cdot \sqrt{\det\{-\nabla^2_{\xi} f(\xi_r)\}} \cdot \frac{1}{\sqrt{(2\pi)^n}} \tag{8.12}$$

where $f(\xi) := R(\xi,\theta) - \lambda\|\xi - \xi_r\|^2$ and $-\nabla^2_{\xi} f(\xi_r)$ is the negative Hessian with respect to ξ.

Since we have assumed that the human's intended trajectory ξ_h is an *improvement* over the robot's trajectory ξ_r, then it must be the case that $R(\xi_h,\theta) > R(\xi_r,\theta)$. Let $\hat{\theta}^0$ be the robot's initial estimate of θ, such that the robot has a prior:

$$P(\theta) = \frac{1}{(2\pi\alpha)^{1/2}} e^{-\frac{1}{2\alpha}\|\theta-\hat{\theta}^0\|^2} \tag{8.13}$$

where α is a positive constant.

Substituting our normalized observation model from Equation (8.12) and the prior from Equation (8.13) back into Equation (8.11), the MAP estimate $\hat{\theta}^{t+1}$ is the solution to:

$$\arg\max_{\theta} \sum_{\tau=0}^{t} \left(R(\xi_h^\tau,\theta) - R(\xi_r^\tau,\theta)\right) - \frac{1}{2\alpha}\|\theta - \hat{\theta}^0\|^2 \tag{8.14}$$

In Equation (8.14) the $\lambda\|\xi_h - \xi_r\|^2$ terms have dropped out because this penalty for human effort does not explicitly depend on θ. For a detailed derivation on the Laplace approximation and the MAP estimate, please see Section 8.10. Intuitively, our estimation problem (8.14) states that we are searching for the objective θ that *maximally separates* the reward associated with ξ_h and ξ_r, while also regulating the size of the change in θ.

We solve Equation (8.14) by taking the gradient with respect to θ and then setting the result equal to zero. Substituting in our cumulative reward function from Equation (8.5), we obtain the *all-at-once* update rule:

$$\begin{aligned} \hat{\theta}^{t+1} &= \hat{\theta}^0 + \alpha \sum_{\tau=0}^{t} \left(\Phi(\xi_h^\tau) - \Phi(\xi_r^\tau)\right) \\ &= \hat{\theta}^t + \alpha\left(\Phi(\xi_h^t) - \Phi(\xi_r^t)\right) \end{aligned} \tag{8.15}$$

Given the current MAP estimate $\hat{\theta}^t$, the robot's trajectory ξ_r^t, and the human's intended trajectory ξ_h^t, we determine an approximate MAP estimate at timestep $t+1$ by comparing the feature counts. Note that the update rule in (8.15) is actually the online gradient descent algorithm [35] applied to our normalized observation model (8.12).

Interpretation. The all-at-once update rule (8.15) has a simple interpretation: if any feature has a higher value along the human's intended trajectory than the robot's trajectory, the robot should increase the weight of that feature. Returning to our example, if the human's preferred trajectory ξ_h moves the coffee closer to the table than the robot's

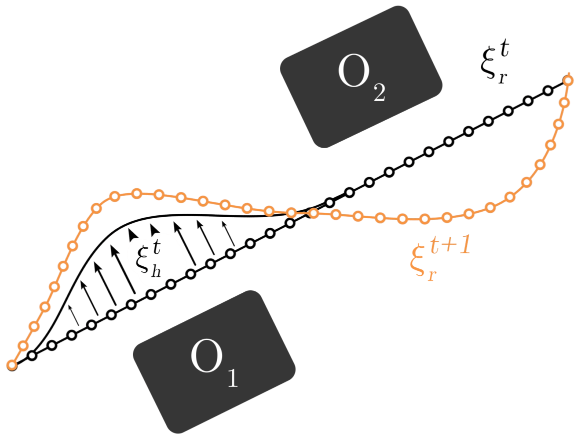


Figure 8.2: Visualization of one iteration of our proposed algorithm for online learning from pHRI. Here a point robot is moving in a 2D environment with two obstacles, O_1 and O_2. The robot initially plans to follow a straight line trajectory from start to goal (ξ_r^t, black dotted line). But the human wants the robot to move farther away from the obstacles: the human pushes the robot, and the robot uses the human's applied force to deform its initial trajectory into a human preferred trajectory (ξ_h^t, solid black line). Given that ξ_h^t is better aligned with the human's objective than ξ_r^t, we compute an online update of θ and replan a new trajectory ξ_r^{t+1} (orange dotted line). Notice that the new trajectory moves the robot farther from the nearby obstacle O_1 and the future obstacle O_2.

original trajectory ξ_r, the weights in $\hat{\theta}$ for distance-to-table will increase. This enables the robot to learn in real-time from corrections.

Interestingly, our all-at-once update rule is a special case of the update rules from two related IRL works. Equation (8.15) is the same as the Preference Perceptron for coactive learning—introduced in [**shivaswamy2015**] and applied for manipulation tasks by [99]—if ξ_h was the robot's original trajectory ξ_r with a single corrected waypoint. Similarly, Equation (8.15) is analogous to online Maximum Margin Planning without the loss function if the correction ξ_h was treated as a new demonstration [175]. These findings also align with work from [54], who show that other IRL methods can be interpreted as a MAP estimate. What is unique in our work is that we demonstrate how the online gradient-descent update rule in Equation (8.15) results from a POMDP with hidden state θ where physical human interactions are interpreted as intended trajectories.

8.5 One-at-a-Time Online Learning

We derived an update rule to learn the human's objective from their physical interactions with the robot. This all-at-once approach changes the weight of *all* the features that the human adjusts during their correction. In practice, however, the human's interactions (and their intended trajectory) may result in *unintended corrections* which mistakenly alter features the human meant to leave untouched. For example, when the human's action intentionally causes ξ_h to move closer to the table, the same correction may accidentally also change the orientation of the coffee. In order to address unintended corrections, we here assume that the human's intended trajectory ξ_h should change only a *single* feature. We explain how to determine which feature the human is trying to change, and then modify the update rule from Equation (8.15) to obtain *one-at-a-time* learning.

Intended Feature Difference. Let us define the change in features at time t as $\Delta\Phi^t = \Phi(\xi_h^t) - \Phi(\xi_r^t) \in \mathbb{R}^N$, where ξ_h^t is the human's intended trajectory, ξ_r^t is the robot's trajectory, and N is the number of features. Given our assumption that the human intends to change just one feature at a single timepoint, $\Delta\Phi^t$ should have only a single non-zero entry; however, because human corrections are imperfect [6, 106] this not always the case. We introduce the *intended feature difference*, $\Delta\Phi_h^t$, where only the feature the human wants to update is non-zero. At each timestep the robot must infer $\Delta\Phi_h^t$ from $\Delta\Phi^t$. Note that this one-at-a-time approach does not mean that only a single feature changes during the entire task: the user can adjust a *different* feature at each timestep.

Without loss of generality, assume the human is trying to change the i-th entry of the robot's MAP estimate $\hat{\theta}$ during the current timestep t. The ideal human correction of ξ_r^t should accordingly change the feature count in the direction:

$$J_i = \frac{\partial \Phi(\xi_r^t)}{\partial \hat{\theta}_i^t} \tag{8.16}$$

Recall that ξ_r^t is optimal with respect to the current estimate $\hat{\theta}^t$, and so changing $\hat{\theta}^t$ will alter $\Phi(\xi_r^t)$. Put another way, if the human is an optimal corrector, and their interaction was meant to alter just the weight on the i-th feature, then we would expect them to correct the current robot trajectory ξ_r^t such that they produce a feature difference $\Delta\Phi^t$ exactly in the direction of the vector J_i from Equation (8.16).

Because the human is imperfect, they will not exactly match Equation (8.16). Instead, we model the human as making corrections $\Delta\Phi^t$ in the direction of J_i. This yields an *observation model* from which the robot can find the likelihood of observing a specific feature difference $\Delta\Phi^t$ given that the human is attempting to update the i-th feature:

$$P(\Delta\Phi \mid i) \propto e^{|J_i \cdot \Delta\Phi|} \tag{8.17}$$

Recalling that the robot observes the feature difference $\Delta\Phi^t = \Phi(\xi_h^t) - \Phi(\xi_r^t)$, then we estimate which feature the human most likely wants to change using:

$$\begin{aligned} i^* &= \arg\max_i P(\Phi(\xi_h^t) - \Phi(\xi_r^t) \mid i) \\ &= \arg\max_i \left| J_i \cdot \left(\Phi(\xi_h^t) - \Phi(\xi_r^t)\right)\right| \end{aligned} \tag{8.18}$$

Once the robot solves for the most likely feature the human wants to change, i^*, it can now find the human's *intended feature difference* $\Delta\Phi_h^t$. Recall that, if the human wanted to only update feature i^*, their intended feature difference would ideally be in the direction J_{i^*}. Thus, we choose $\Delta\Phi_h^t \propto J_{i^*}$ as our intended feature difference.

Update Rule. We make two simplifications to derive a one-at-a-time update rule. Both simplifications stem from the difficulty of evaluating the partial derivative from Equation (8.16) in real-time. Indeed, rather than computing this partial derivative, we approximate J_i as proportional to the vector $(0, \ldots, 1, \ldots, 0)$, where the i-th entry is non-zero. Intuitively, we are here assuming that when the i-th weight in $\hat{\theta}$ changes, it predominately induces a change in the i-th feature along the resulting optimal trajectory.

Given this assumption, computing the intended feature difference $\Delta\Phi_h^t \propto J_{i^*}$ reduces to projecting the observed feature difference $\Delta\Phi^t$ induced by the human's action u_h onto the i^*-th axis:

$$\Delta\Phi_h^t = (0, \ldots, \Delta\Phi_{i^*}^t, \ldots, 0) \tag{8.19}$$

This fulfills our original requirement for the intended feature difference $\Delta\Phi_h^t$ to only have one non-zero entry. Moreover, once we substitute our simplification of J_i back into our feature estimation problem (8.18), we get a simple yet intuitive heuristic for finding i^*: only the feature which the user has changed the *most* during their correction should be updated. Our *one-at-a-time* update rule is therefore similar to the gradient update from Equation (8.15), but with a single feature weight update using Equation (8.19):

$$\hat{\theta}^{t+1} = \hat{\theta}^t + \alpha\Delta\Phi_h^t \tag{8.20}$$

Instead of updating the estimated weights associated with all the features like in Equation (8.15), we now only update the MAP estimate for the feature which has the largest change in feature count. Overall, isolating a single feature at every timestep is meant to mitigate the effects of unintended learning from noisy physical interactions[5].

8.6 Optimally Responding to pHRI

Before introducing all-at-once and one-at-a-time learning, we showed how approximate solutions to pHRI involve (a) safely tracking the optimal trajectory and (b) updating the MAP estimate based on human interactions. Now that we have derived update rules for $\hat{\theta}$, we will circle back and present our algorithm for learning from pHRI. We also include practical considerations for implementation.

Algorithm. We have formalized pHRI as an instance of a POMDP and then approximated that POMDP as a QMDP. To solve this QMDP we must both find the robot's optimal policy and update the MAP estimate of θ at every timestep t. First, we approximate the robot's optimal policy by solving a trajectory optimization problem in Equation (8.6) for ξ_r^t and then tracking ξ_r^t with an impedance controller (8.8). Second, we update the MAP estimate

[5]We note that all the features are normalized to have the same sensitivity.

Algorithm 1 Online Learning from pHRI

```
Given: initial weights $\hat{\theta}^0$ and features $\phi \in [0,1]^N$
Initialize: $\xi_r^0 \leftarrow \arg\max_\xi \hat{\theta}^0 \cdot \Phi(\xi)$
for $t = 0$ to $T$ do
    $u_r^t = B_r(\dot{q}_r^t - \dot{q}^t) + K_r(q_r^t - q^t)$        ▷ (8.8)
    $\xi_h^t \leftarrow \xi_r^t + \mu A^{-1} U_h^t$        ▷ (8.9)
    $\hat{\theta}^{t+1} \leftarrow \hat{\theta}^t + \alpha\big(\Phi(\xi_h^t) - \Phi(\xi_r^t)\big)$        ▷ (8.15) or (8.20)
    $\xi_r^{t+1} \leftarrow \arg\max_\xi \hat{\theta}^{t+1} \cdot \Phi(\xi)$        ▷ (8.6)
end for
```

$\hat{\theta}^t$ by interpreting each human correction as an intended trajectory—which we obtain by deforming the robot's original trajectory using Equation (8.9)—and next we perform either all-at-once (8.15) or one-at-a-time (8.20) online updates to obtain $\hat{\theta}^{t+1}$. At the next timestep $t+1$ the robot replans its optimal trajectory under $\hat{\theta}^{t+1}$ and the process repeats. An overview is provided in Algorithm 1.

Implementation. In practice, Algorithm 1 uses impedance control to track a trajectory that is replanned after pHRI. We note, however, that this approach ultimately derives from formulating pHRI as a POMDP. One possible variation on this algorithm is—instead of replanning ξ_r^t from start to goal—replanning ξ_r^t from the robot's current state x^t to the goal. The advantage of this variation is that it saves us the time of recomputing the trajectory before our current state (which the robot does not need to know). However, in our implementation we always replan from start to goal. This is because constantly setting x^t along the desired trajectory prevents the human from experiencing any impedance during interactions (i.e., the robot never resists the human's interactions). Without any haptic feedback from the robot, the end-user cannot easily infer the current robot's trajectory, and so the human does not know whether additional corrections are necessary [101]. A second consideration deals with the robot's feature space. Throughout this work we assume that the robot knows the relevant features ϕ, which are provided by the robot designer or user [11]. Alternatively, the robot could use techniques like feature selection [86] to filter a set of available features, or the features could be learned by the robot [136].

8.7 Simulations

To compare our real-time learning approach with optimal offline solutions and current online baselines, as well as to test both all-at-once and one-at-a-time learning, we conduct human-robot interaction simulations in a controlled environment. Here the robot is performing a pick-and-place task: the robot is carrying a cup of coffee for the simulated human. The simulated human physically interacts with the robot to correct its behavior.

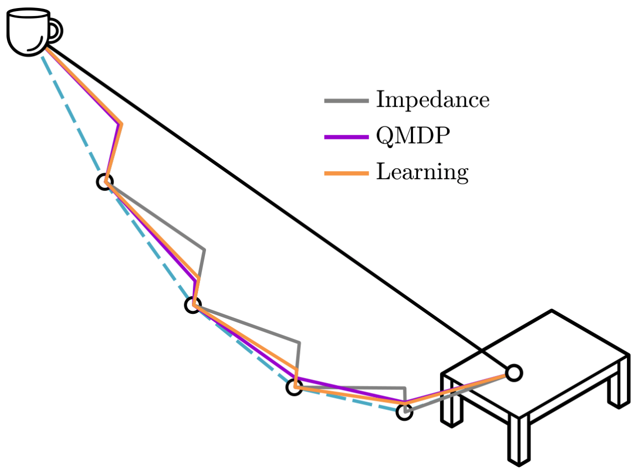


Figure 8.3: Comparison of the offline QMDP solution and our online Learning approximation for a pick-and-place task. The robot is attempting to carry the cup to the table. Originally the robot is confident it should move in a straight line (black), but the user actually wants the cup to be carried closer to the ground (blue, dashed). Here the human physically interacts to guide the robot back to their desired trajectory (circles) when the robot's error is too high.

Setup. We perform three separate simulated experiments. In each, the robot is moving within a planar world from a fixed start position to a fixed goal position. We here use a 2-DoF *point robot* for simplicity, while noting that we will use a 7-DoF robotic manipulator during our user studies. The robot's state is $x \in \mathbb{R}^2$, the robot's action is $u_r \in \mathbb{R}^2$, and the human's action is $u_h \in \mathbb{R}^2$; both the state and action spaces are continuous. We assume that the robot knows the relevant features ϕ, but the robot does not know the human's objective θ. The robot initially believes that "velocity" (i.e., trajectory length) is the only important feature, and so the robot tries to move in a *straight line* from start to goal.

Learning vs. QMDP vs. No Learning. To learn in real-time, we introduced several approximations on top of separating estimation from control (QMDP). Here we want to assess how much these approximations reduce the robot's performance. We first compare our approximate real-time solution described in Algorithm 1 to the complete QMDP solution [142]. As a baseline, we have also included just using impedance control [87], where no learning takes places from the humans interactions. Thus, the three tested approaches are *Impedance*, *QMDP*, and *Learning*. The simulated task is depicted in Fig. 8.3. The two features are "velocity" and "table," and the human wants the robot to carry their coffee closer to table level ($\theta = 1$). During each timestep, if the robot's position error from the human's desired trajectory exceeds a predefined threshold, then the human physically corrects the robot by guiding it to their desired trajectory. Recall that our Learning method uses a MAP estimate of the human's objective, but the full QMDP solution maintains a belief b over θ. For QMDP simulations, we discretize the belief space—such that $\theta \in \{0, 1\}$—and the robot starts with a prior $b^0(\theta = 1) = 0.1$. Using

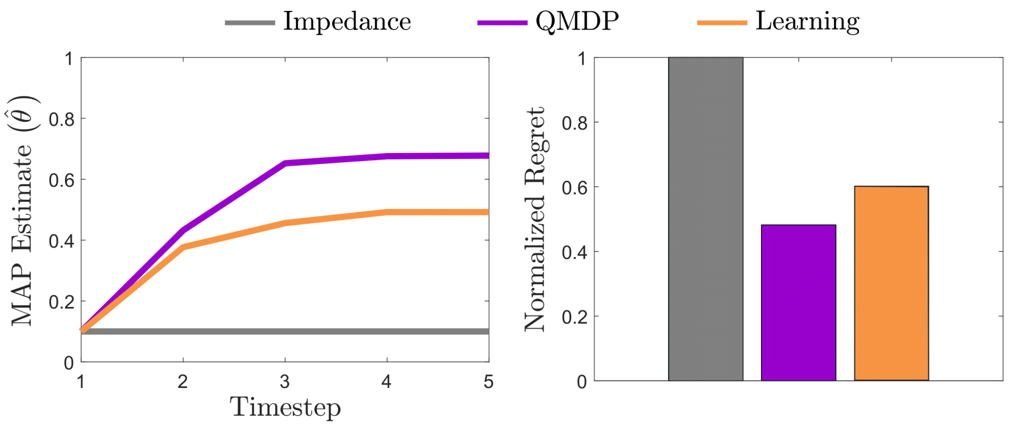


Figure 8.4: Robot learning and regret for the task from Fig. 8.3. The true human objective is $\theta = 1$. The offline QMDP solution learns more about the human's objective than our online Learning approximation. However, both QMDP and Learning lead to significantly less regret than the Impedance baseline. The regret for QMDP is the lowest because here human corrects the robot at one less timestep.

a planar environment and a discretized belief space enables us to actually compare the full QMDP solution to our approximation, since the QMDP becomes prohibitively expensive in high dimensions with continuous state, action, and belief spaces.

We expect the full QMDP solution to outperform our Learning approximation. From Fig. 8.4, we observe that the robot learns θ faster when using the QMDP, and that the robot completes the task with less regret. Both QMDP and Learning outperform Impedance, where the robot does not learn from pHRI. We note that here the simulated human behaves differently than our observation model (8.2): rather than maximizing their Q-value, the human is guiding the robot along their desired trajectory. When the simulated human *does* follow our observation model, we obtain very similar results: the normalized regret becomes 0.55 for QMDP and 0.62 for Learning. To ensure that the learning rate is consistent between the QMDP and Learning methods, we selected α such that $\hat{\theta}^1$ equalled $b^1(\theta = 1)$ when the simulated human followed our observation model (8.2). From these simulations we conclude that the Learning approximation for online performance is worse than the full QMDP solution, but the difference between these methods is *negligible* when compared to Impedance.

Learning vs. Deforming. As part of our approximations we assumed that the human's interaction implies an *intended trajectory*. Here we want to see whether learning from the intended trajectory—as in Algorithm 1—is more optimal than simply setting that intended trajectory as the robot's trajectory. We compare two real-time learning methods: our *Learning* approach, and the trajectory deformation method from [146], which we refer to as *Deforming*. The task used in these simulations is shown in Figs 8.5 and 8.6. Again, the robot is carrying a cup of coffee, but here the human would prefer for the robot to avoid carrying this coffee over their laptop. Thus, the two features are "velocity" and

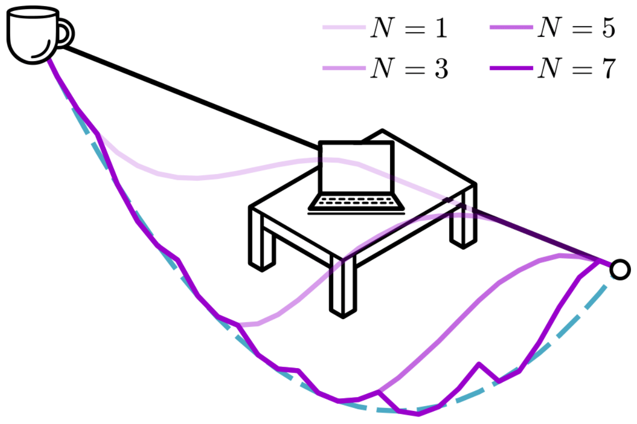


Figure 8.5: Responding to physical interaction by deforming the robot's trajectory. We propagate the human's interaction along the robot's trajectory to get ξ_h, the human's intended trajectory. We then set ξ_h as the robot's trajectory. Here N is the number of number of interactions: we show the robot's trajectory after 1, 3, 5, and 7 deformations. Importantly, when using deformations the robot never learns about task, but only updates its trajectory in the direction of the human's applied force.

"laptop." As before, the simulated human corrects the robot by guiding it back to their desired trajectory when the tracking error exceeds a predefined limit. In Deforming the robot does not learn about the human's objective, but instead propagates the human's corrections along the rest of the robot's trajectory. By contrast, in Learning we treat these trajectory deformations as the human's intended trajectory, which is then leveraged in our online update rule. Learning and Deforming can both be applied to change the robot's desired trajectory in real-time in response to pHRI, and Deforming is the same as treating the intended trajectory as the robot's trajectory.

In Figs. 8.5 and 8.6 we show the robot's trajectory after N human corrections. Notice that Deformations result in local changes which aggregate over time, while—when we learn from these deformations—Learning replans the entire trajectory. Our findings are summarized in Fig 8.7: it takes fewer corrections to track the human's desired trajectory with Learning, and the human also expends more effort with Learning. To make the comparison consistent, here we used the same propagation method from (8.9) to get the Deformations and the intended trajectory for Learning. Based on our results, we conclude that Learning leads to more efficient online performance than Deformations alone, and, in particular, Learning requires *less human effort* to complete the task correctly.

All-at-Once vs. One-at-a-Time. Previously we simulated tasks with only two features, and so a single feature weight was sufficient to capture the human's preference ($\theta \in \mathbb{R}$). In other words, either the all-at-once update or the one-at-a-time update could have been used for Learning. Now we compare *All-at-Once* (8.15) and *One-at-a-Time* (8.20) learning in a task with three features ($\theta \in \mathbb{R}^2$). This task is illustrated in Figs. 8.8 and 8.10. The human end-user trades off between the length of the robot's trajectory (velocity), the

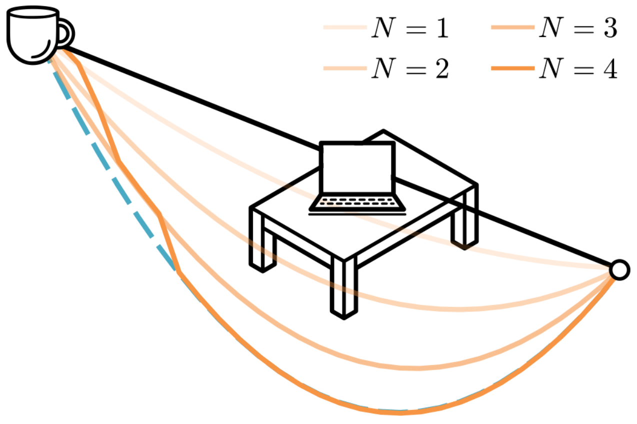


Figure 8.6: Responding to physical interactions using our proposed learning approach. As before, we propagate the human's interaction along the robot's current trajectory to get ξ_h, the human's intended trajectory. But now we go one step further: we compare ξ_h to ξ_r to update our estimate of the human's objective θ. The robot then moves in the direction of the optimal trajectory for θ. Under this approach the robot learns to avoid the laptop after $N = 4$ corrections, and autonomously tracks the human's preferred trajectory (blue, dashed).

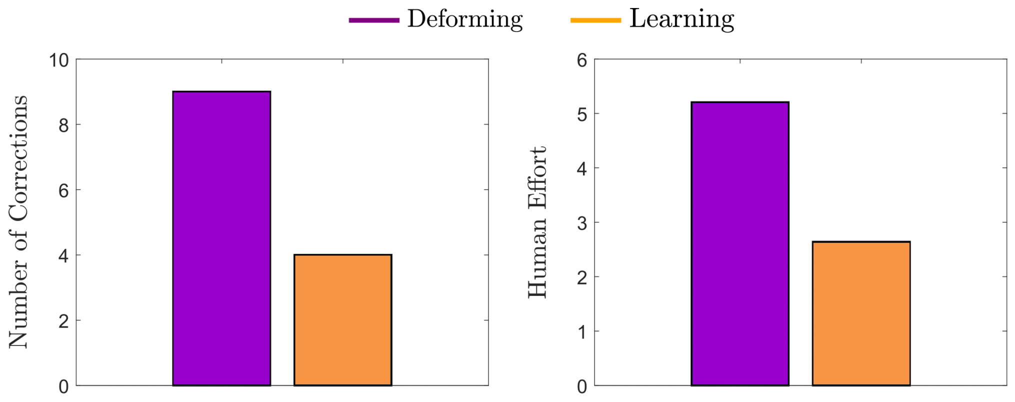


Figure 8.7: Comparison of Deforming and Learning across our simulations in Figs. 8.5 and 8.6. When robots only deform their trajectory in the direction of the human's applied force, humans must exert more effort and make more corrections to guide the robot's trajectory to their desired behavior. By contrast, Learning from these deformations enables the robot to correct not only the next few timesteps, but also to replan the remainder of the trajectory based on the human's correction.

coffee's height above the table (table), and the robot's distance from the person (human). Like before, the weight associated with "velocity" is fixed, and the human's true objective is $\theta = [0.5, 0]$, where 0.5 is the weight associated with table and 0 is the weight associated with human. Initially the robot believes that $\theta^0 = [0, 0]$, and therefore the robot is unaware that it should move closer to the table.

We utilize two different simulated humans: (a) an *optimal human*, who exactly guides

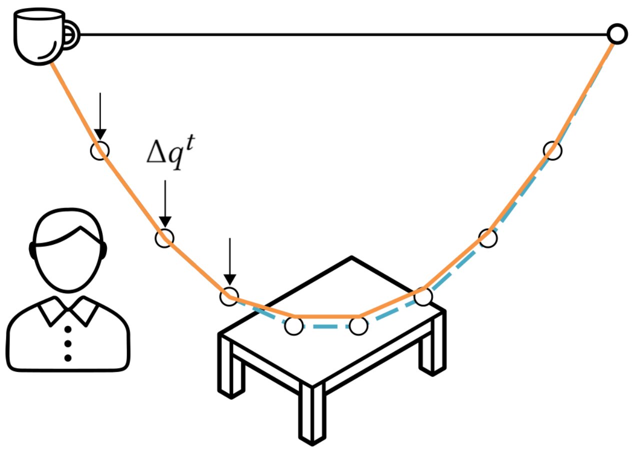


Figure 8.8: Comparing All-at-Once and One-at-a-Time learning with an *optimal* simulated human. This human wants the robot to carry the coffee closer to table level, and provides physical corrections that exactly match their preferences. The human corrects the robot's behavior over the first few timesteps (arrows) and the robot autonomously follows the human's desired trajectory after these corrections. The robot's behavior is the same for All-at-Once and One-at-a-Time learning.

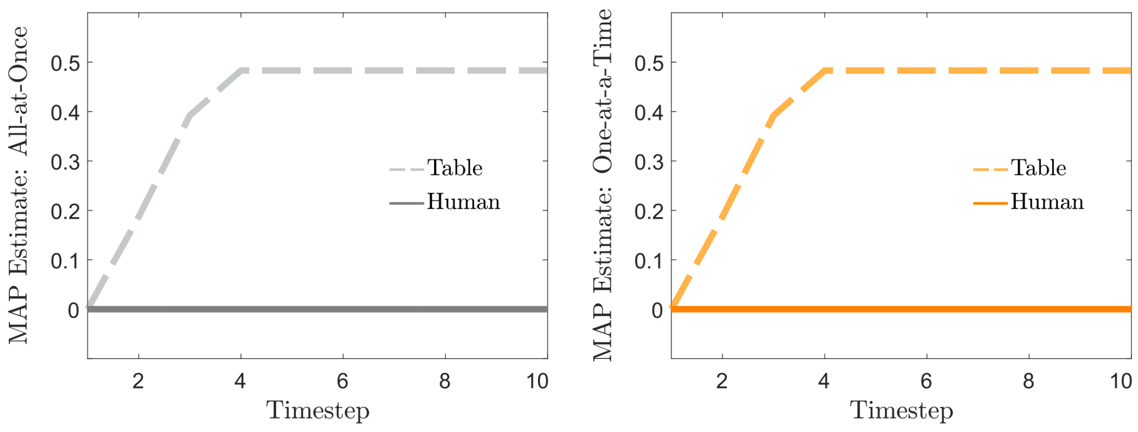


Figure 8.9: All-at-Once and One-at-a-Time learning with an *optimal* simulated human. The true objective is $table = 0.5$, $human = 0$. Both All-at-Once and One-at-a-Time converge to the true objective: no unintentional corrections occur.

the robot towards their desired trajectory, and (b) a *noisy human*, who imperfectly corrects the robot's trajectory. Like in our previous simulations, the human intervenes to correct the robot when the robot's error with respect to their desired trajectory exceeds an acceptable margin of error: let us now refer to this as the optimal human. By contrast, the noisy human takes actions sampled from a Gaussian distribution which is centered at the optimal human's action. This distribution is biased in the direction of the human such that the noisy human tends to accidentally pull the robot closer to their body when correcting the

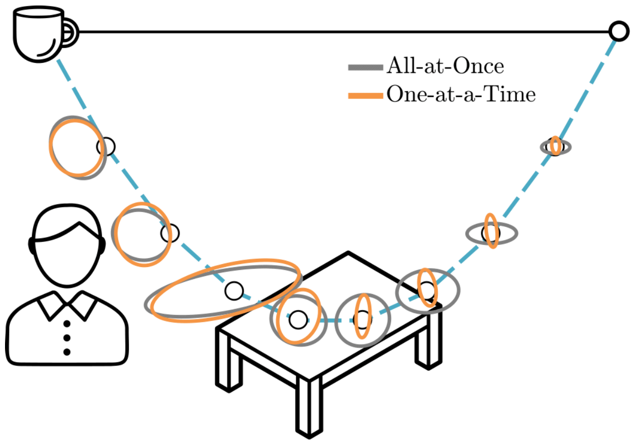


Figure 8.10: Comparing All-at-Once and One-at-a-Time learning with a *noisy* simulated human. This noisy human wants the robot to move closer to the table, but accidentally provides biased corrections that also move the cup closer to the human. Ellipses show the robot's position at each timestep with 95% confidence over 100 simulations. The human unintentionally pulls the robot closer to their body at the start of the task, and with the All-at-Once approach they struggle to undo these mistakes in the second half of the task.

table feature. Due to this noise and bias, the noisy human may *unintentionally* correct the human feature.

Our final simulation compares All-at-Once and One-at-a-Time learning for optimal and noisy humans. The results for an optimal human are shown in Figs. 8.8 and 8.9, while the results for the noisy human are depicted in Figs. 8.10 and 8.11. We find that the performance of All-at-Once and One-at-a-Time are identical when the human acts optimally: the robot accurately learns the importance of table, and does not change the weight of human. When the person acts noisily, however, One-at-a-Time learning causes better performance. More specifically, the noisy user corrected the All-at-Once robot during an average of 5.24 timesteps, but only corrected the One-at-a-Time robot 3.56 timesteps. Inspecting Fig. 8.11, we observe that the noisy human unintentionally taught the human feature at the beginning of the task, and had to exert additional effort undoing this mistake on All-at-Once robots. We conclude that there is a benefit to One-at-a-Time learning when the human behaves noisily, since updating only one feature per timestep *mitigates accidental learning*.

8.8 User Studies

To evaluate the benefits of using physical interaction to communicate we conducted two user studies with a 7-DoF robotic arm (JACO2, Kinova). In the first study, we tested *whether* learning from pHRI is useful when humans interact, and compared our online learning approach to a state-of-the-art response that treated interactions as disturbances

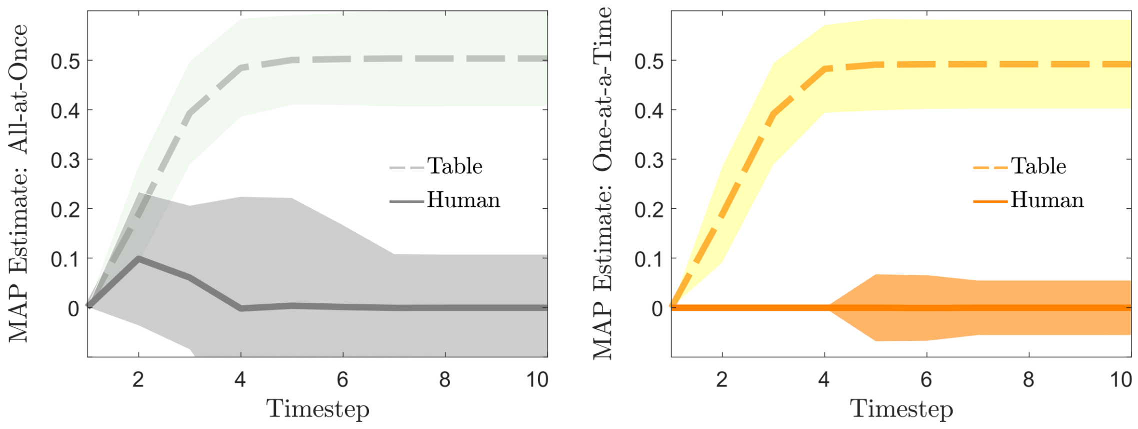
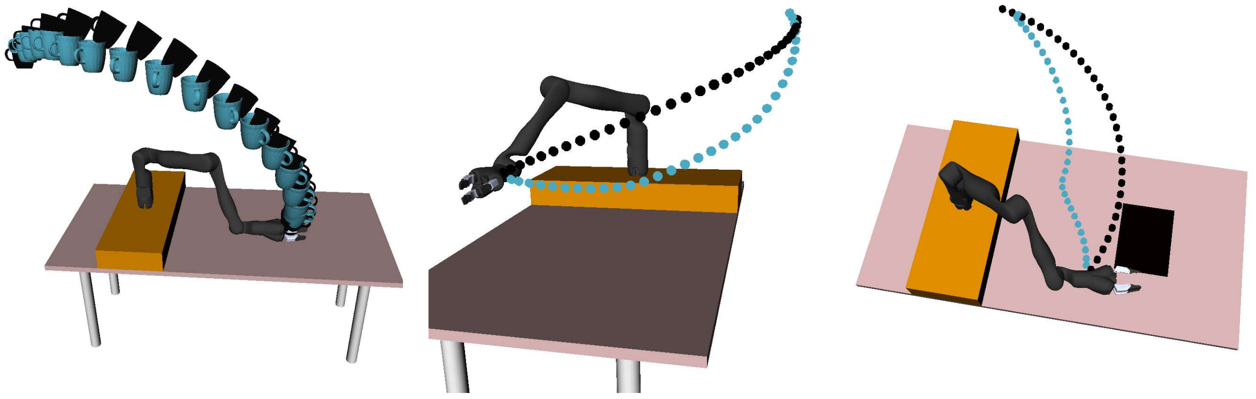


Figure 8.11: All-at-Once and One-at-a-Time learning with a *noisy* simulated human. The shaded regions give the standard error of the mean. With All-at-Once, the robot initially learns that the human feature is important, and the person must undo that unintended learning. One-at-at-Time learning reduces the unintended effects of the human's noisy corrections; the robot converges towards the human's desired trajectory more rapidly.



(a) Task 1: Keep the cup upright (b) Task 2: Carry closer to the table (c) Task 3: Avoid the region above a laptop

Figure 8.12: Simulations depicting the robot trajectories for each of the three tasks in our first user study (Learning vs. Impedance). The black path represents the robot's initial trajectory, and the blue path represents the human's desired trajectory.

(Learning vs. Impedance). In the second study, we tested *how* the robot should learn from end-users, and compared one-at-a-time learning to all-at-once learning (One-at-a-Time vs. All-at-Once). During both studies the participants and the robot worked in close physical proximity. In all experimental tasks, the robot began with the wrong objective function, and participants were instructed to physically interact with the robot to correct

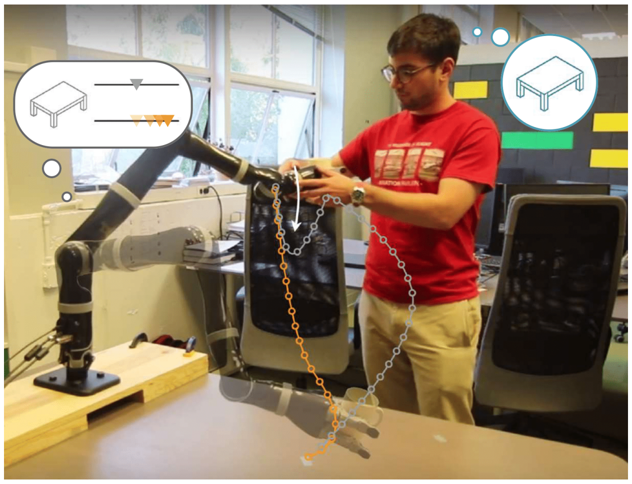


Figure 8.13: During the first user study participants interacted with a robot that maintained a fixed objective (Impedance, grey) and a robot that learned from their physical interactions to update its objective (Learning, orange).

its behavior[6].

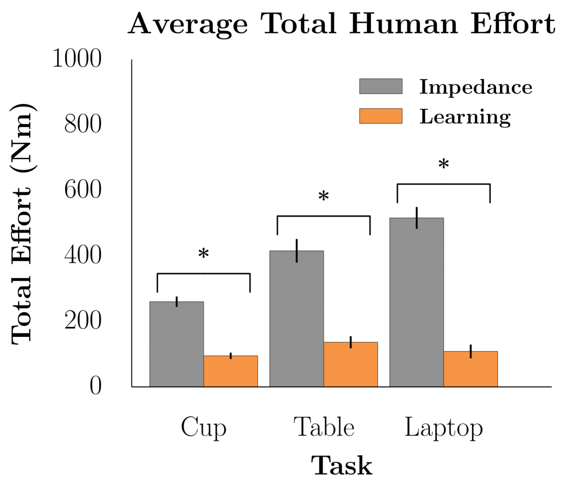
8.8.1 Learning vs. Impedance

We have argued that pHRI is a means for humans to correct the robot's behavior. In our first user study, we compare a robot that treats human interactions as intentional (and learns from them) to a robot that assumes all human interactions are disturbances (and ignores them).

Independent Variables. We manipulated the *pHRI strategy* with two levels: *Learning* and *Impedance*. The Learning robot used our proposed method (Algorithm 1) to react to physical corrections and re-plan a new trajectory during the task. By contrast, the Impedance robot used impedance control (our method without updating $\hat{\theta}$) to react to physical interactions and then return to the originally planned trajectory. Because impedance control is currently the most common strategy for responding to pHRI [87], we treated Impedance as the state-of-the-art.

Dependent Measures. We measured the robot's objective performance with respect to the human's actual objective. One challenge in designing our experiment was that each participant might have a different internal objective θ for any given task depending on their experiences and preferences. Since we did not have direct access to every person's internal preferences, we defined the true objective θ ourselves, and conveyed the objective to participants by demonstrating the desired optimal robot behavior. We instructed participants to correct the robot to achieve this behavior with as little interaction as possible.

[6]For video footage of the experiment, see: https://youtu.be/I2YHT3giwcY



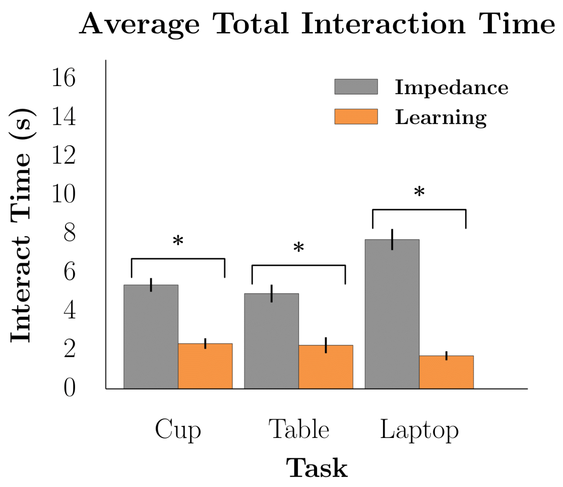
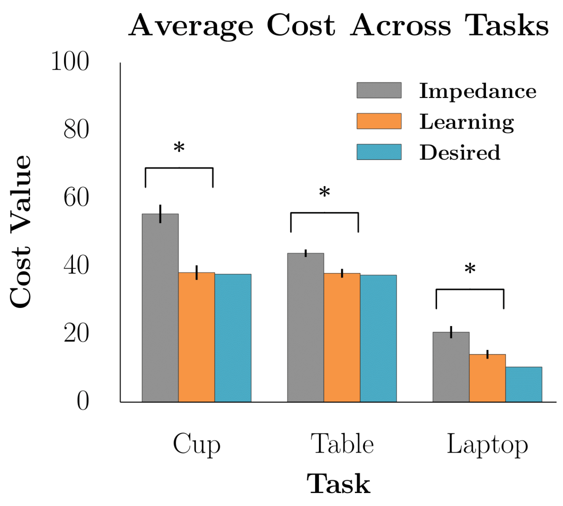


Figure 8.14: Objective results from our first user study. We explored whether robots should learn from physical interactions (Learning vs Impedance). Learning from pHRI decreased participant effort and interaction time across all experimental tasks (the total trajectory time was 15s). An asterisk (*) means $p < .0001$.



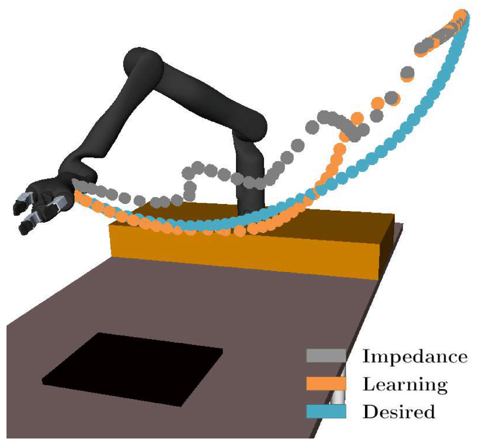


Figure 8.15: (Left) Average cost for each task and the cost of the desired trajectory. Robots that always follow the human's desired trajectory minimize cost. An asterisk (*) means $p < 0.0001$. (Right) Plot of sample participant data from the laptop task: the desired trajectory is in blue, the trajectory with the Impedance condition is in gray, and the Learning condition trajectory is in orange.

To understand how users perceived the robot, we also asked subjects to complete a 7-point Likert scale survey for both pHRI strategies: the questions from this survey are shown in Table 8.1.

Hypotheses.

H1. *Learning will decrease interaction time, effort, and cumulative trajectory cost.*

H2. *Learning users will believe the robot understood their preferences, feel that interacting with the robot was easier, and perceive the robot as more predictable and collaborative.*

Tasks. We designed three household manipulation tasks for the robot to perform in a shared workspace, in addition to one familiarization task. The robot's objective function consisted of two features: "velocity" and a task-specific feature, where $\Phi(\xi) \in [0, 1]$. Because one feature weight was sufficient to capture these tasks (i.e., $\theta \in \mathbb{R}$) both the all-at-once and one-at-a-time learning approached were here identical. For each task, the robot carried a cup from a start pose to a goal pose with an *initially incorrect objective*, forcing participants to correct its behavior during the task.

In the familiarization task the robot's original trajectory moved too close to the human. Participants had to physically interact with the robot to make the robot keep the cup farther away from their body. In Task 1 the robot carried a cup directly from start to goal, but did not realize that it needed to keep this cup upright. Participants had to intervene to prevent the cup from spilling. In Task 2 the robot carried the cup too high in the air, risking breaking that cup if it were to slip. Participants had to correct the robot to keep the cup closer to the table. Finally, in Task 3 the robot moved the cup over a laptop to reach its final goal pose, and participants physically guided the robot away from this laptop region. We include a depiction of our three experimental tasks in Fig. 8.12.

Participants. We employed a within-subjects design and counterbalanced the order of the pHRI strategy conditions. Ten total members of the UC Berkeley community (5 male, 5 female, age range 18-34) provided informed consent according to the approved IRB protocol and participated in the study. All participants had technical backgrounds. None of the participants had prior experience interacting with the robot used in our experiments.

Procedure. For each pHRI strategy participants performed the familiarization task, followed by the three experimental tasks, and then filled out our user survey. They attempted every task twice during each pHRI strategy for robustness (we recorded the attempt number for our analysis). Since we artificially set the true objective θ, we showed participants both the original and desired robot trajectory before the task started to make sure that they understood this objective and got a sense of the corrections they would need to make.

Results – Objective. We conducted a repeated measures ANOVA with pHRI strategy (Impedance or Learning) and trial number (first attempt or second attempt) as factors. We applied this ANOVA to three objective metrics: total participant effort, interaction time, and cost[7]. Fig. 8.14 shows the results for human effort and interaction time, and Fig. 8.15 shows the results for cost. Learning resulted in significantly less interaction force ($F(1, 116) = 86.29, p < 0.0001$) interaction time ($F(1, 116) = 75.52, p < 0.0001$), and task cost ($F(1, 116) = 21.85, p < 0.0001$). Interestingly, while trial number did not significantly

[7]For simplicity, we only measured the value of the feature that needed to be modified in each task, and computed the absolute difference from the feature value of the optimal trajectory.

	Questions	Cronbach's α	Imped LSM	Learn LSM	F(1,9)	p-value
understanding	By the end, the robot understood how I wanted it to do the task. Even by the end, the robot still did not know how I wanted it to do the task. The robot learned from my corrections. The robot did not understand what I was trying to accomplish.	0.94	1.70	5.10	118.56	**<.0001**
effort	I had to keep correcting the robot. The robot required minimal correction.	0.98	1.25	5.10	85.25	**<.0001**
predict	It was easy to anticipate how the robot will respond to my corrections.	0.8	4.90	4.70	0.06	0.82
	The robot's response to my corrections was surprising.	0.8	3.10	3.70	0.89	0.37
collab	The robot worked with me to complete the task. The robot did not collaborate with me to complete the task.	0.98	1.80	4.80	55.86	**<.0001**

Table 8.1: Subjective ratings collected from a 7-point Likert scale survey. Participants answered each question once after working with the Impedance condition, and once after the Learning condition. The four question scales are shown on the left. Imped is short for Impedance, Learn is short for Learning, and LSM stands for Likert scale mean. Higher LSM values are better (more understanding, less effort, more predictable, more collaborative). ANOVA results are on the far right.

affect participant's performance with either method, attempting the task a second time yielded a marginal improvement for the impedance strategy but not for the learning strategy. This may suggest that it is easier for users to familiarize themselves with the impedance strategy.

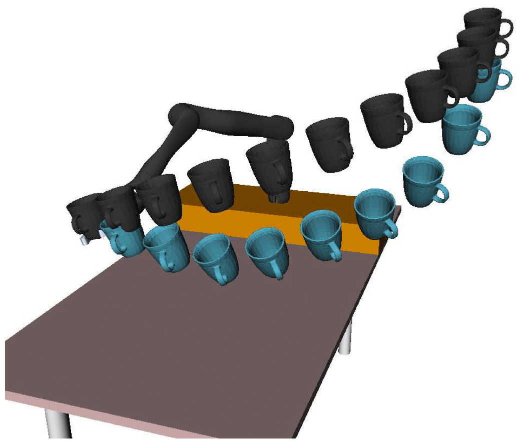
Overall, our results support **H1**. Using interaction forces to learn about the objective θ here enabled the robot to better complete its tasks with less human effort when compared to a state-of-the-art impedance controller.

Results – Subjective. Table 8.1 shows the results of our participant survey. We tested the reliability of four scales, and found the understanding, effort, and collaboration scales to be reliable. Thus, we grouped each of these scales into a combined score, and ran a one-way repeated measures ANOVA on each resulting score. We found that the robot using our Learning method was perceived as significantly ($p < 0.0001$) more understanding, less difficult to interact with, and more collaborative than the Impedance approach.

By contrast, we found no significant difference between our Learning method and the baseline Impedance method in terms of predictability. Participant comments suggest that while the robot quickly adapted to their corrections when Learning (e.g. "the robot seemed to quickly figure out what I cared about and kept doing it on its own"), determining what the robot was doing during Learning was less intuitive (e.g. "if I pushed it hard enough sometimes it would seem to fall into another mode, and then do things correctly").

We conclude that **H2** was partially supported: although users did not perceive Learning to be more predictable than Impedance, participants believed that the Learning robot understood their preferences better, took less effort to interact with, and was a more collaborative partner.

Summary. Robots that treat pHRI as a source of information (rather than as a disturbance) are capable of online, in-task learning. Learning from pHRI resulted in better objective and subjective performance than a traditional Impedance approach. We found that the Learning robot better matched the human's preferred behavior with less human effort and interaction time, and participants perceived the Learning robot as easier to understand and collaborate with. However, participants did not think that the Learning robot was more predictable than the Impedance robot.


(a) Task 1: Correct one feature, the distance to table (table)

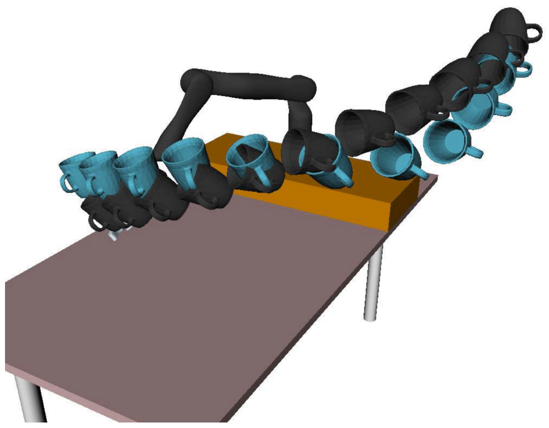

(b) Task 2: Correct two features, the cup orientation (cup) and the distance to table (table)

Figure 8.16: Simulations depicting the robot trajectories for both of the two tasks in our second user study (One-at-a-Time vs. All-at-Once). The black path represents the robot's original trajectory, and the blue path represents the human's desired trajectory. Note that the robot now has multiple features, making it possible for the human to accidentally correct one or both features.

8.8.2 One-at-a-Time vs. All-at-Once

We have found that learning from pHRI is beneficial; now we want to determine *how* the robot should learn. In our second user study we focused on objective functions which encode multiple task-related features. In these scenarios it is difficult for the robot to determine which aspects of the task the person meant to correct during pHRI, and which features were changed unintentionally.

Independent Variables. We used a 2-by-2 factorial design and manipulated the *learning strategy* with two levels (*All-at-Once* and *One-at-a-Time*), as well as the *number of feature weights that need correction* (one feature weight and all the feature weights). Within the All-at-Once learning strategy the robot always updated all the feature weights after a single human interaction using the gradient update from Equation (8.15). In the One-at-a-Time condition the robot chose the one feature that changed the most using Equation (8.18), and then updated its feature weight according to Equation (8.20). Both learning strategies leveraged Algorithm 1, but with different update rules. By comparing these two versions of our approach we explore how robots should respond to noisy and imperfect human interactions.

Dependent Measures – Objective. Within this user study the robot carried a cup across a table. To analyze the objective performance of our two learning strategies, we split the objective measures into four categories:

Final Learned Reward: These metrics measure how closely the learned reward matched the optimal reward by the end of the task (timestep T). We measured the dot product between the optimal and final reward vector: $DotFinal = \theta \cdot \hat{\theta}^T$. We also analyzed the regret of the final learned reward, which is the weighted feature difference between the ideal trajectory and the learned trajectory:

$$RegretFinal = \theta \cdot \Phi(\xi_\theta) - \theta \cdot \Phi(\xi_{\hat{\theta}^T})$$

Lastly, we measured the individual feature differences (table and cup) between the ideal and final learned trajectories:

$$TableDiffFinal = |\Phi_{table}(\xi_\theta) - \Phi_{table}(\xi_{\hat{\theta}^T})|$$

$$CupDiffFinal = |\Phi_{cup}(\xi_\theta) - \Phi_{cup}(\xi_{\hat{\theta}^T})|$$

Learning Process: Measures about the learning process, i.e., $\theta = \{\hat{\theta}^0, \hat{\theta}^1, \dots, \hat{\theta}^T\}$, included the average dot product between the true reward and the estimated reward over time:

$$DotAvg = \frac{1}{T}\sum_{i=0}^{T} \theta \cdot \hat{\theta}^i$$

We also measured the length of the $\tilde{\theta}$ path through weight space for both cup ($\tilde{\theta}_{cup}$) and table ($\tilde{\theta}_{table}$) weights. Finally, we computed the number of times the cup and table weights were updated in the opposite direction of the optimal θ (denoted by *CupAway* and *TableAway*).

Executed Trajectory: For the actual trajectory that the robot executed, ξ_{act}, we measured the regret

$$Regret = \theta \cdot \Phi(\xi_\theta) - \theta \cdot \Phi(\xi_{act})$$

and the individual table and cup feature differences between the ideal and actual trajectory

$$TableDiff = |\Phi_{table}(\xi_\theta) - \Phi_{table}(\xi_{act})|$$

$$CupDiff = |\Phi_{cup}(\xi_\theta) - \Phi_{cup}(\xi_{act})|$$

Interaction: Interaction measures on the forces applied by the human included the total interaction force, $IactForce = \sum_{t=0}^{T} ||u_h^t||_1$, and the total interaction time.

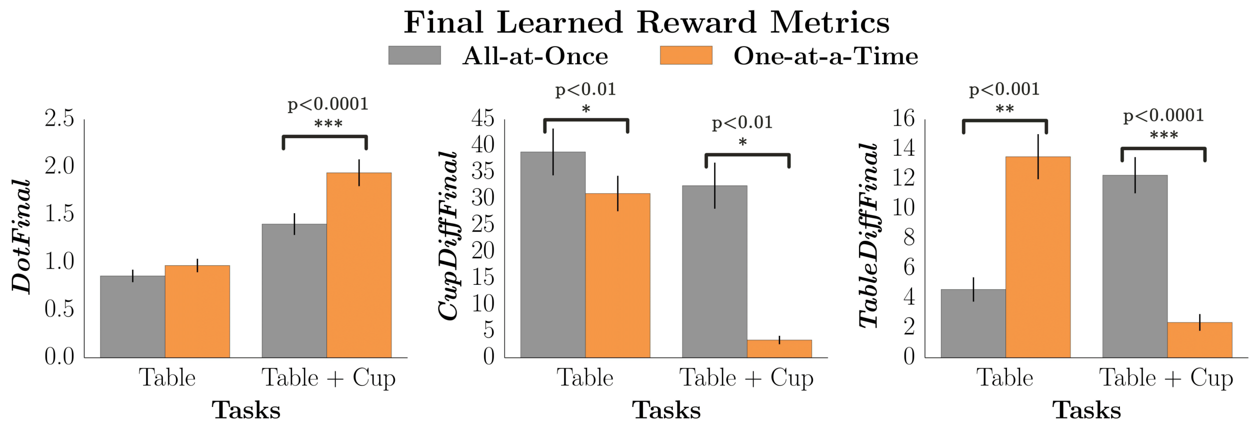


Figure 8.17: How accurately the robot learned when using All-at-Once or One-at-a-Time. (Left) The final learned θ with One-at-a-Time is more aligned with the ideal θ on the Table+Cup task where the human had to correct multiple features. Looking at the individual feature errors: (Center) while the final cup feature was closer to ideal for One-at-a-Time on both tasks, (Right) All-at-Once learned a more accurate estimate of Table when the human only needed to teach a single feature. But we notice an interaction effect here: although One-at-a-Time got the Table wrong on the single feature task, it outperformed All-at-Once across the board when the human needed to adjust multiple features.

Dependent Measures – Subjective. After each of the four conditions we administered a 7-point Likert scale survey about the participant's interaction experience (see Table 8.2 for the list of questions). We separated our survey items into four scales: success in teaching the robot about the task (succ), correctness of update (correct update), needing to undo corrections because the robot learned something wrong (undoing), and ease of undoing (undo ease).

Hypotheses.

> **H3.** *One-at-a-Time learning will increase the final learned reward, enable a better learning process, result in lower regret for the executed trajectory, and lead to less interaction effort and time as compared to All-at-Once.*
>
> **H4.** *Participants will perceive the robot as more successful at accomplishing the task, better at learning, less likely to need undoing, and easier to correct if it did learn something wrong in the One-at-a-Time condition.*

Tasks. We designed two household manipulation tasks for the robot arm to perform within a shared workspace. A depiction of the these experimental tasks is shown in Fig. 8.16. The robot's objective function consisted of three features: "velocity," (the trajectory length), "table" (the distance from the table), and "cup" (the orientation of the cup). We purposely selected features that were easy for participants to interpret so that they intuitively understood how to correct the robot. For each experimental task the robot carried a cup from a start pose to end pose with an *initially incorrect objective*. Task 1 focused on participants having to correct a *single aspect* of the objective, while Task 2 required them to correct *all parts* of the objective.

In Task 1 the robot's objective had only *one feature weight incorrect*. The robot's default trajectory took a cup from the participant and put it down on the table, but carried the cup too far above the table (see top of Fig. 8.16). In Task 2 *all the feature weights started out incorrect* in the robot's objective. The robot again took a cup from the participant and put it down on the table, but this time it initially grasped the cup at the wrong angle, and was also carrying the cup too high above the table (see bottom of Fig. 8.16).

Participants. We used a within-subjects design and counterbalanced the order of the conditions during experiments. In total, twelve members of the UC Berkeley community (4 male, 7 female, 1 non-binary trans-masculine, age range 18-30) provided informed written consent according to the approved IRB protocol before participating in this study. Eleven of the participants had technical backgrounds, and one did not. None of the participants had prior experience interacting with the robot used in our experiments.

Procedure. Before the start of the experiment participants performed a familiarization task to become more comfortable teaching the 7-DoF JACO2 robot with physical corrections. We here used the second task from our first experiment, where the robot carried a cup at an angle, and the human must correct the cup's orientation. During this familiarization task the robot's objective contained only one feature weight (cup). Afterwards, for each experimental task, the participants were shown the robot's initial trajectory as well as their desired trajectory. They were also told what aspects of the task the robot is aware of (cup orientation and distance to table), as well as which learning strategy they were interacting with (One-at-a-Time or All-at-Once). Participants were told the difference between the two learning strategies in order to minimize in-task learning effects. Importantly, we did *not* tell participants to teach the robot in any specific way (like one aspect as a time); we only informed participants about how the robot reasons over their corrections.

Results – Objective. Here we summarize the results for each of our objective dependent measures.

Final Learned Reward. We ran a factorial repeated-measures ANOVA with learning strategy and number of features as factors—and user ID as a random effect—for each of our objective metrics. Fig. 8.17 summarizes our findings about the final learned weights $\hat{\theta}^T$ for both learning strategies.

For the final dot product with the true reward θ, we found a significant main effect of the learning strategy ($F(1, 81) = 29.86$, $p < .0001$), but also an interaction effect with the number of features ($F(1, 81) = 13.07$, $p < .01$). The post-hoc analysis with Tukey HSD revealed that One-at-a-Time led to a higher dot product on Task 2 ($p < .0001$), but there was no significant difference on Task 1 (where One-at-a-Time led to slightly higher dot product).

We next looked at the final regret, i.e., the difference between the cost of the final learned trajectory and the cost of the ideal trajectory. For this metric we found an interaction effect, suggesting that One-at-a-Time led to lower regret for Task 2 but not for Task 1. Looking separately at the feature values for table and cup, we found that One-at-a-Time led to a significantly lower difference for the cup feature across the board ($F(1, 81) = 11.30, p < .01$, no interaction effect), but that One-at-a-Time only improved the difference for the table on Task 2 ($p < .0001$). Surprisingly, One-at-a-Time significantly increased the difference when the human only needed to correct a single feature ($p < .001$).

Overall, we see that One-at-a-Time results in better final learning when the human needs to correct multiple features (Task 2). When the human only wants to correct a single feature (Task 1) the results are mixed: One-at-a-Time led to a significantly better result for the cup orientation, but a significantly worse result for the table distance.

Learning Process. For the average dot product between the estimated and true reward over time, our analysis revealed almost identical outcomes as those reported for the final reward (see Fig. 8.18). Higher values of $DotAvg$ indicate the robot's estimate $\hat{\theta}$ is in the direction of the true parameters θ. Differences in $DotAvg$ were negligible during Task 1, but One-at-a-Time outperformed All-at-Once during Task 2.

Next, we found that One-at-a-Time resulted in significantly fewer updates in the wrong direction for the cup weight ($F(1, 81) = 44.91, p < .0001$) and for the table weight ($F(1, 81) = 22.02, p < .0001$), with no interaction effect in either case. Fig. 8.19 highlights these findings and their connection to the subjective user responses from Table 8.2 that are related to undoing.

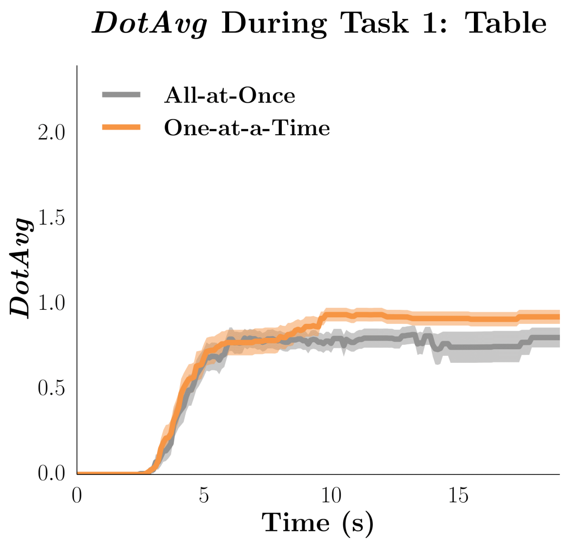
Finally, looking at the length of the learned path $\tilde{\theta}$ through the space of feature weights, we found a main effect of learning strategy ($F(1, 81) = 26.82, p < .0001$), but also an interaction effect ($F(1, 81) = 6.55$, $p = .01$). The post-hoc analysis with Tukey HSD revealed that for Task 1 our One-at-a-Time approach resulted in a significantly shorter path through weight space ($p < .0001$). The path was also shorter during Task 2, but this difference was not significant. The effect was mainly due to the One-at-a-Time method resulting in a shorter path for the cup weight on Task 1, as revealed by the post-hoc analysis ($p < .0001$).

Overall, we see that the quality of the learning process was significantly higher for the One-at-a-Time strategy across both tasks. When one aspect (Task 1) or all aspects (Task 2) of the objective were wrong, One-at-a-Time led to fewer weight updates in the wrong direction, and resulted in the learned reward over time being closer to the true reward.

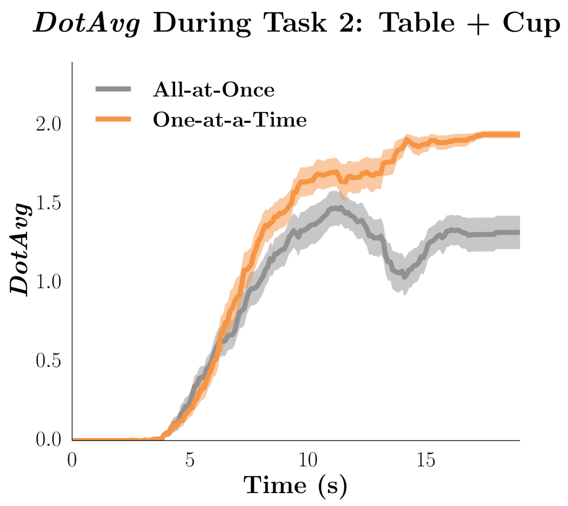
The Executed Trajectory. We found no significant main effect of the learning strategy on the regret of the executed trajectory: the two strategies lead to relatively similar actual trajectories with respect to regret. Both regret as well as the feature differences from ideal for cup and table showed significant interaction effects.

Interaction Metrics. We found no significant effects on interaction time or force.

Objective Results – Summary. Taken together these results indicate that One-at-a-Time leads to a better overall learning process. On the more complex task where all the features



(a) DotAvg measures the alignment between the learned $\hat{\theta}^t$ and the true θ. In the task with only one wrong feature weight, there was no significant difference between the two methods in average dot product over time.



(b) In contrast to (a), when two feature weights are wrong One-at-a-Time outperformed All-at-Once. The dip in DotAvg for All-at-Once indicates that participants accidentally taught the robot the wrong thing and needed to undo their corrections.

Figure 8.18: One-at-a-Time showed more consistent alignment between the learned objective, $\hat{\theta}^t$, and the ideal objective, θ, when compared to All-at-Once. Contrasting (a) and (b), these results suggest that when the human needs to correct multiple aspects of the robot's behavior One-at-a-Time enables more accurate learning. We anticipate that most real-world tasks will require corrections of multiple features.

must be corrected (Task 2), One-at-a-Time also leads to a better final learned reward. For the simpler task where only one feature must be corrected (Task 1), One-at-a-Time enables users to better avoid accidentally changing the initially correct weight (cup), but One-at-a-Time is not as good as the All-at-Once method at enabling users to properly correct the initially incorrect weight (table). Accordingly, our objective results partially support **H3**. Although updating one feature weight at a time does not improve task performance when only one aspect of the objective is wrong, reasoning about one feature weight at a time leads to significantly better learning and task performance when all aspects of the objective are wrong.

Results – Subjective. We ran a repeated measures ANOVA on the results of our participant survey. After testing the reliability of our four scales (see Table 8.2), we found that the correct update and undoing scales were reliable, and so we grouped these into a combined score. The success (succ) scale had only a single question, and so grouping was not applicable here. Finally, we analyzed the two questions related to undoing ease (undo ease) individually because this specific scale was not reliable.

For the correct update scale we found a significant effect of learning strategy ($F(1, 33) =$

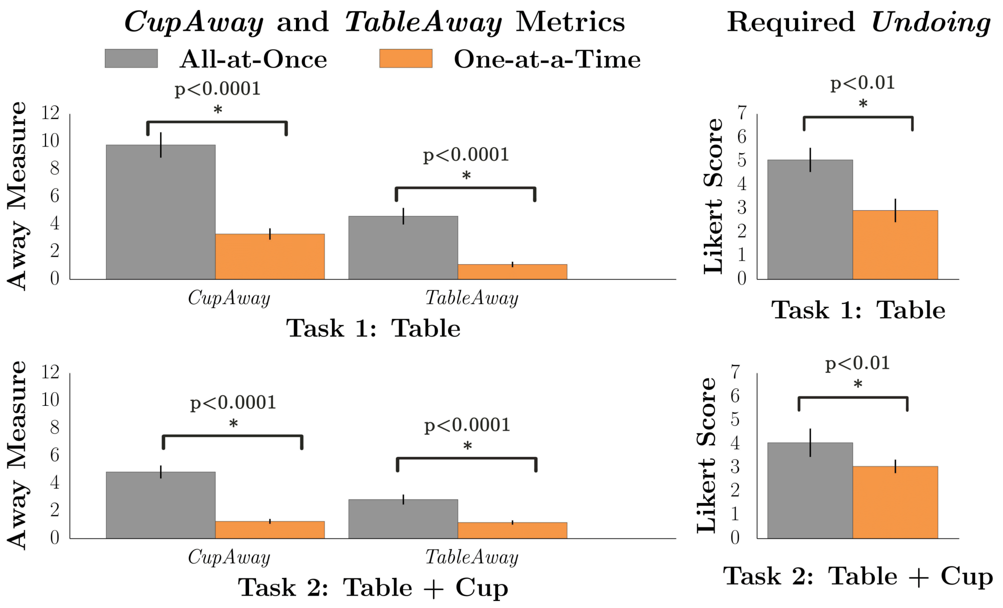


Figure 8.19: How frequently participants made mistakes and had to undo their corrections. (Left) Humans working with One-at-a-Time made fewer corrections that caused the robot to learn the opposite of what they intended. This result was consistent across both tasks. (Right) These objective findings match our subjective Likert scale data. Participants thought the One-at-a-Time robot was less likely to learn the wrong thing and need an additional undoing action.

5.09, $p = 0.031$), showing that participants perceived One-at-a-Time as better at updating the robot's objective according to their corrections. The undoing scale also showed a significant effect of learning strategy ($F(1, 33) = 10.35, p < 0.01$), where One-at-a-Time was perceived as less likely to learn the wrong thing, which would then force the participants to undo their corrections. For both success and undoing ease scales we analyzed the questions Q1, Q9, and Q10 individually and found no significant effect of learning strategy.

Subjective Results – Summary. The subjective data echoes some of our objective data results. Participants perceived that the robot with One-at-a-Time was better at correcting what they intended, and required less undoing due to unintended learning. We conclude that **H4** was partially supported.

8.9 Discussion

In this work we recognize that when humans physically interact with and correct a robot's behavior their corrections become a source of information. This insight enables us to formulate pHRI as a partially observable dynamical system: the robot is unsure of its true objective function, and human interactions become observations about that latent objective. Solving this dynamical system results in robots that respond to pHRI in

	Likert Questions	Cronbach's α
succ	Q1: I successfully taught the robot how to do the task.	–
correct update	Q2: The robot correctly updated its understanding about aspects of the task that I did want to change.	.84
	Q3: The robot wrongly updated its understanding about aspects of the task I did NOT want to change.	
	Q4: The robot understood which aspects of the task I wanted to change, and how to change them.	
	Q5: The robot misinterpreted my corrections.	
undoing	Q6: I had to try to undo corrections that I gave to the robot, because it learned the wrong thing.	.93
	Q7: Sometimes my corrections were just meant to fix the effect of previous corrections I gave.	
	Q8: I had to re-teach the robot about an aspect of the task that it started off knowing well.	
undo ease	Q9: When the robot learned something wrong, it was difficult for me to undo that.	.66
	Q10: It was easy to re-correct the robot whenever it misunderstood a previous correction of mine.	

Table 8.2: Likert scale questions from our user study comparing All-at-Once and One-at-a-Time. Questions were grouped into four categories: success in accomplishing the task (succ), whether the robot's update was what the human wanted (correct update), how often the human needing to undo corrections because of unintended learning (undoing), and how easy it was to undo a mistake (undo ease).

the *optimal way*. These robots update their understanding of the task after each human interaction, and then change how they complete the rest of the current task based on this new understanding.

Approximations. Directly applying our formalism to find the robot's optimal response to pHRI is generally not tractable in high-dimensional and continuous state and action spaces. We therefore derive an online approximation for robot learning and control. We first leverage the QMDP approximation [142] to separate the learning problem from the control problem, and then move from the policy level to the trajectory level. This results in two local optimization problems. In the first, the robot solves for an optimal trajectory given its MAP estimate of the task objective, and then tracks that trajectory using impedance control [97]. The second optimization problem occurs at timesteps when the human interacts: here the robot updates its estimate of the correct objective using online gradient descent [35]; this update rule is a special case of Coactive Learning [99, 197] and Maximum Margin Planning [175]. Although we can practically think of the proposed algorithm as using impedance control to track a trajectory that is replanned after physical interactions, this approach ultimately derives from formulating pHRI as an instance of a POMDP.

Interestingly, this derivation enables us to interpret other state-of-the-art responses to pHRI as simplifications of our approximation. For example, if the robot never updates its estimate of the correct objective function (i.e., the robot never learns from pHRI), then our online approximation reduces to impedance control. Alternatively, if we treat the intended trajectory induced by the human's correction as the robot's trajectory (but do not update the robot's objective), then our approximation reduces to deforming the desired trajectory [146]. We compared our online approximation to both of these simplifications—impedance control and deformations—as well as to a more complete QMPD solution. During offline simulations we found that the performance loss between our learning method and the QMPD policy was negligible, but our method outperformed impedance control and trajectory deformations. During user studies with a 7-DoF robot, our learning approach resulted in decreased interaction time, effort, and cumulative trajectory cost when compared to an impedance controller. We also found that users believed the learning robot better understood their preferences, resulted in less interaction effort, and was more a collaborative partner than the impedance robot.

Unintended Corrections. While we assert that the human's physical interactions are often intentional, we also recognize that physical interactions are inherently noisy and imperfect. When correcting a high DoF robot the human may adjust aspects of the robot's behavior that they did not intend to. If the robot treats every aspect of the human's correction as intentional this can result in unintended learning, which the human must then undo with additional corrections. In order to mitigate the effects of unintended corrections, and make the process of correcting robots through pHRI more intuitive for the end-user, we introduce a restriction to our online learning rule. More specifically, we assume that the robot should only learn about one aspect of the task from each human correction. During offline simulations we showed that this One-at-a-Time learning approach outperformed All-at-Once when the simulated user acted noisily: with All-at-Once, the noisy human unintentionally changed aspects of the robot's task which were already correct, but with Once-at-a-Time these unintended corrections were avoided.

Next, we performed a user study to compare our One-at-a-Time and All-at-Once learning strategies. Here the robot could reason over multiple features during two tasks: one task required correcting a single feature, and the other task required correcting multiple features of the robot's objective. For the multiple feature task learning about one feature at a time was objectively superior: it led to a better final learning outcome, took a shorter path to the optimum, and had fewer incorrect inferences and human undoing along the way. But the results were not as clear for the single feature task: One-at-a-Time reduced unintended learning on the weights that were initially correct, but it hindered learning for the initially incorrect weights. Overall, study participants subjectively preferred One-at-a-Time to All-at-Once: they thought One-at-a-Time was better at learning the intended aspects of their corrections and required less undoing.

Based on these results, we hypothesize that the superior objective performance of One-at-a-Time was due to the increased complexity of the teaching task. It appears that only learning a single aspect at a time is more useful when the teaching task becomes more complex and requires that the human alter multiple parts of the robot's objective. When the teaching task is simple, however, and only requires one aspect of the objective to change, it is not yet clear whether One-at-a-Time is a better learning strategy.

Limitations. Our work is a step towards understanding how robots should respond to pHRI. When selecting the approximations for online learning, as well as the method for inferring which feature to update in One-at-a-Time, we opt for approximations that are consistent to those in the existing literature. Future work and hardware advances may remove the need for some of the approximations we have leveraged.

Throughout our paper we assumed that the robot had access to the necessary task-related features. Moreover, during our user studies the robot's objective contained only two or three total features, and these features were intuitive to the human (e.g., "distance-to-person"). In practice objective functions will have larger features sets and may include task-related features that are non-intuitive to the human: additional work is needed to investigate how well our learning strategies perform in these cases.

Finally, solutions that can handle dynamical aspects—like preferences about the timing of the robot's trajectory—would require a different approach for inferring the intended human trajectory. Here it may actually be necessary to return from the trajectory space to the policy space.

8.10 Detailed Derivation: Laplace Approximation & MAP

Recall our observation model from (8.10):

$$P(\xi_h \mid \xi_r, \theta) = \frac{e^{R(\xi_h,\theta)-\lambda||\xi_h-\xi_r||^2}}{\int e^{R(\xi,\theta)-\lambda||\xi-\xi_r||^2} d\xi}. \tag{8.21}$$

The key challenge with using this observation model is computing the denominator, since computing an integral over the space of all trajectories is computationally intractable. Thus, we are interested in approximating the denominator. To do this, we will perform a Laplace approximation[152], which locally models the distribution as a Gaussian which then allows us to obtain a solution for the integral in closed form.

Let's just focus on the denominator:

$$\int e^{R(\xi,\theta)-\lambda||\xi-\xi_r||^2} d\xi \tag{8.22}$$

Step 1. "Quadraticize" the exponent. Recall that for a function $f : \mathbb{R}^n \longrightarrow \mathbb{R}^m$, the 2nd order TSE around $a \in \mathbb{R}^n$ is:

$$f(x) \approx f(a) + \nabla_x f(a)^\top (x-a) + \frac{1}{2}(x-a)^\top \nabla_x^2 f(a)(x-a)$$

For notational simplicity, let our function of interest (i.e. the objective function) be:

$$f(\xi) := R(\xi,\theta) - \lambda||\xi-\xi_r||^2$$

For our approximation, we will *assume that the robot's planned trajectory $\xi_r \in \mathbb{R}^n$ is locally optimal*. We will do a TSE around ξ_r to obtain:

$$f(\xi) \approx R(\xi_r,\theta) + \nabla_\xi f(\xi_r)^\top (\xi-\xi_r) + \frac{1}{2}(\xi-\xi_r)^\top \nabla_\xi^2 f(\xi_r)(\xi-\xi_r) \tag{8.23}$$

$$= R(\xi_r,\theta) + \frac{1}{2}(\xi-\xi_r)^\top \nabla_\xi^2 f(\xi_r)(\xi-\xi_r) \qquad \textit{(}\nabla_\xi f(\xi_r) = 0 \textit{ at optimum)} \tag{8.24}$$

$$= R(\xi_r,\theta) - \Big(-\frac{1}{2}(\xi-\xi_r)^\top \nabla_\xi^2 f(\xi_r)(\xi-\xi_r)\Big) \quad \textit{(match Gaussian form)} \tag{8.25}$$

Step 2. Approximate the denominator. We will now approximate the denominator as an unnormalized Gaussian, using our 2nd order TSE:

$$\int e^{R(\xi,\theta)-\lambda||\xi-\xi_r||^2} d\xi \approx \int e^{R(\xi_r,\theta)-\left(-\frac{1}{2}(\xi-\xi_r)^\top \nabla_\xi^2 f(\xi_r)(\xi-\xi_r)\right)} d\xi \tag{8.26}$$

$$= e^{R(\xi_r,\theta)} \int e^{-\left(-\frac{1}{2}(\xi-\xi_r)^\top \nabla_\xi^2 f(\xi_r)(\xi-\xi_r)\right)} d\xi \tag{8.27}$$

$$= e^{R(\xi_r,\theta)} \frac{\sqrt{(2\pi)^n}}{\sqrt{\det\{-\nabla_\xi^2 f(\xi_r)\}}} \qquad \textit{(apply Gaussian integral soln.)} \tag{8.28}$$

where n is the dimension of the trajectory.

Step 3. Plug into original probability distribution. Going back to our original probability distribution, we now can obtain an approximate, but simpler closed-form solution:

$$P(\xi_h \mid \xi_r, \theta) = \frac{e^{R(\xi_h,\theta)-\lambda||\xi_h-\xi_r||^2}}{\int e^{R(\xi,\theta)-\lambda||\xi-\xi_r||^2} d\xi} \tag{8.29}$$

$$\approx \frac{e^{R(\xi_h,\theta)-\lambda||\xi_h-\xi_r||^2}}{e^{R(\xi_r,\theta)} \frac{\sqrt{(2\pi)^n}}{\sqrt{\det\{-\nabla^2_\xi f(\xi_r)\}}}} \tag{8.30}$$

$$= \left(e^{R(\xi_h,\theta)-R(\xi_r,\theta)-\lambda||\xi_h-\xi_r||^2}\right) \cdot \sqrt{\det\{-\nabla^2_\xi f(\xi_r)\}} \cdot \frac{1}{\sqrt{(2\pi)^n}} \tag{8.31}$$

Step 4. Simplifying the MAP estimate. Ultimately, we seek to use this probability distribution to obtain the maximum a posteriori (MAP) estimate. For simplicity of exposition, assume we have just one observation. Our MAP estimate can be simplified using our approximated observation model:

$$\hat{\theta} = \arg\max_\theta P(\xi_h \mid \xi_r, \theta) P(\theta) \tag{8.32}$$

$$= \arg\max_\theta \ln P(\xi_h \mid \xi_r, \theta) + \ln P(\theta) \qquad \textit{(log likelihood)} \tag{8.33}$$

$$\approx \arg\max_\theta R(\xi_h, \theta) - R(\xi_r, \theta) - \lambda||\xi_h - \xi_r||^2 + \tag{8.34}$$

$$\ln \sqrt{\det\{-\nabla^2_\xi f(\xi_r)\}} + \ln \frac{1}{\sqrt{(2\pi)^n}} + \ln P(\theta) \quad \textit{(plug in approx. \& simplify)} \tag{8.35}$$

$$= \arg\max_\theta R(\xi_h, \theta) - R(\xi_r, \theta) + \frac{1}{2} \ln \det\{-\nabla^2_\xi f(\xi_r)\} + \ln P(\theta) \qquad \textit{(remove constant terms)} \tag{8.36}$$

$$= \arg\max_\theta \theta^\top \Big(\Phi(\xi_h) - \Phi(\xi_r)\Big) + \frac{1}{2} \ln \det\{-\nabla^2_\xi f(\xi_r)\} + \ln P(\theta) \qquad \textit{(plug in (8.5))} \tag{8.37}$$

Two things worth noting:

- If $f(\xi)$ is quadratic[8] in ξ, then we know that the hessian $\nabla^2_\xi f$ will be constant w.r.t ξ.
- While we wrote f as a function of ξ for notational simplicity throughout, f in fact depends on both the trajectory ξ, and the reward parameter θ. This means that in theory the Hessian term (even if constant w.r.t. ξ) still could contribute to the maximization of θ when computing the MAP.

[8]In our case where $f(\xi) = \theta^\top \Phi(\xi) - \lambda||\xi - \xi_r||^2$, we need $\Phi(\xi)$ to be quadratic in ξ.

In this derivation, we make the simplifying assumptions that (1) the objective f is quadratic[9] in ξ, and (2) the first term, $\theta^\top\Big(\Phi(\xi_h) - \Phi(\xi_r)\Big)$, which looks at the difference in rewards along the original robot trajectory and the induced human trajectory, will dominate[10] the second term involving the Hessian and drop it from the optimization as a final approximation. Thus, we arrive at:

$$\hat{\theta} = \arg\max_\theta P(\xi_h \mid \xi_r, \theta)P(\theta) \approx \arg\max_\theta \theta^\top\Big(\Phi(\xi_h) - \Phi(\xi_r)\Big) + \ln P(\theta) \tag{8.38}$$

This derivation is straightforward to adapt to the scenario where the MAP inference utilizes multiple observations.

8.11 Conclusion

In this work we present an online, in-task response to pHRI that treats human interactions as intentional. We first formulate the problem of responding to pHRI as a partially observable dynamical system, where solving this system defines the *optimal way* for the robot to react. Unfortunately, this formalism is not directly applicable because we require online solutions in high-dimensional and continuous state, action, and belief spaces. We therefore derive an approximate solution for real-time learning and control. During offline simulations we compared our approximate learning method to a complete solution and state-of-the-art baselines, which are actually simplifications of our approach. We perform two separate user studies on a 7-DoF robot arm to determine (a) whether learning from pHRI is useful and (b) how the robot should learn from physical human interactions. While these simulations and user studies indicate the benefits of our approach, we recognize that this work is only a first step towards leveraging the implicit communication present during human-robot interactions.

[9] This is similar to the approximation made by Dragan and Srinivasa in [65].

[10] This requires the second derivatives of $\Phi(\xi)$ to be small, which may not always hold. For an alternative algorithm for computing the MAP where the Hessian is preserved, see [135].

Chapter 9

Quantifying Hypothesis Space Misspecification

This chapter is based on the paper "Quantifying Hypothesis Space Misspecification in Learning from Human-Robot Demonstrations and Physical Corrections" [34] written in collaboration with Andreea Bobu, Jaime Fisac, Sampada Deglurkar, and Anca Dragan.

Autonomous systems are increasingly interfacing and collaborating with humans in a variety of contexts, such as semi-autonomous driving, automated control schemes on airplanes, or household robots working in close proximity with people. While the improving capabilities of robotic systems are opening the door to new application domains, the substantially greater complexity and interactivity of these settings makes it challenging for system designers to account for all relevant operating conditions and requirements ahead of time. For example, a household robot designer may not know how an end-user would like the robot to interact with the personal possessions in the user's home.

In situations like these, it can be beneficial for the robot to utilize human input as guidance on the desired behavior. In fact, human input has enabled researchers and engineers to program advanced behaviors that would have otherwise been extremely challenging to specify. Helicopter acrobatics [2], aggressive automated car maneuvers [121], and indoor navigation [128] are three cases that exemplify the benefit of using human input for guiding robot behavior.

In order to utilize human input, system designers typically equip robots with a representation of possible objectives that the human *could* care about. These representations can range from quadratic cost models[122] to complex temporal logic specifications [79] to neural networks [70]. However, anticipating all motivations for human input and specifying a complete model is challenging. Consider Fig. 9.1 where a human is attempting to change the robot's behavior in order to make it consistently stay close to the table, but the robot's model of what the human might care about does not include distances to the table. By choosing a class of functions, the system designer implicitly assumes that what the human wants (and is giving input about) can be represented via a member of that class.

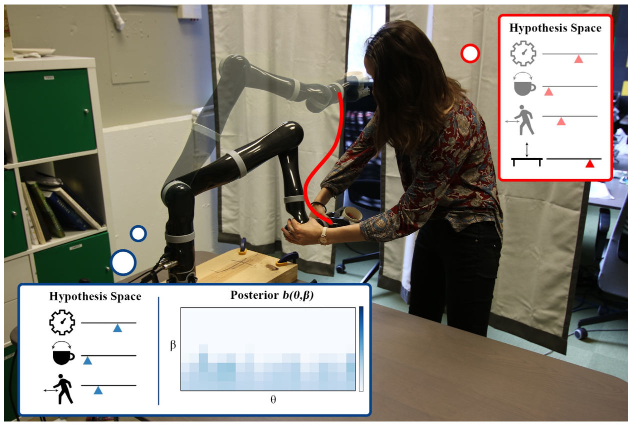


Figure 9.1: A household robotics scenario where the person physically interacts with the robot. The person prefers the robot to keep cups closer to the table, but accounting for the table (outside of collisions) is not in the robot's hypothesis space for what the person might care about. Thus, the robot's internal situational confidence, β, about what the human input means is low for all hypotheses θ.

Unfortunately, when this assumption breaks, the system can misinterpret human guidance, perform unexpected or undesired behavior, and degrade in overall performance.

Two approaches to mitigating this problem could be to either start with a more complex objective space or to continuously increase its complexity given more data. Unfortunately, even complex models are not guaranteed to encompass all possibilities and re-computing the best objective space based on human data faces the threat of overfitting to the most recent observations. In contrast, we argue the robot should be able to *understand when it cannot understand the input*. For example, if the end-user in the home is trying to guide the robot to handle fragile objects with care but the system does not posses a model of fragility, the robot should deduce that this input cannot be well explained by any of its given hypotheses.

In this work, we formalize how autonomous systems can explicitly reason about how well they can explain given human inputs. To do this, we observe that if a human input appears unlikely with respect to all possible hypotheses, then the robot's model is misspecified. We build on previous work centered around this observation to propose a Bayesian inference framework focused on inferring both model parameters, and their corresponding *situational confidence*. If the robot is in situations like Fig. 9.1 where none of the hypotheses explain the human's input well, then the situational confidence will be low for all hypotheses, indicating that the robot's model is not sufficiently rich to understand the human's input. However, when the robot's model is well specified, our framework does not impede the robot from inferring the correct task objectives — in fact, the situational confidence will be high, providing an indicator of how well the system can understand the objective.

We illustrate the utility of situational confidence estimation in quantifying objective space misspecification for two types of human input: demonstrations and corrections.

Our contributions in this work are:

1. we introduce a general framework for quantifying objective space misspecification when the human and the robot are acting on the same dynamical system;

2. we showcase the framework for learning from demonstrations using user demonstration data for an arm motion planning task;

3. we showcase the framework for learning from physical corrections by deriving an algorithm for online (close to real-time) inference and testing it in a user study.

We note that this work is an extension of [33], which was originally presented at the Conference on Robot Learning, 2018. We build on this work by introducing a general framework for quantifying objective space misspecification, and instantiating it in a new type of human input: learning from demonstrations. Not only are demonstrations the most widely used type of input for learning objective functions, but the applicability across two input types suggests that the approach could be adapted more broadly to more types of human feedback.

The remainder of this paper is organized as follows: Section 9.1 places this work in the context of existing literature on robots learning from humans and model confidence estimation. Section 9.2 frames the confidence estimation problem more formally for scenarios where the human and robot operate on the same dynamical system. Section 9.3 directly instantiates the framework in Section 9.2 for the case of learning from demonstrations. Section 9.4 presents a derivation of approximations of the general formalism for tractable online inference from human corrections. Section 9.5 showcases our proposed approach in several case studies where the robot's hypothesis space cannot or only partially explain the human's input. Section 9.6 presents the results of a user study of our approach as applied to a 7-DoF robotic manipulator learning from human participants. Section 9.7 concludes with a discussion of some of the limitations of our work, as well as suggestions for future research directions.

Overall, we think that the ability to detect misspecification when learning objectives from human input will become increasingly important as robotics capability advances and we will want end-users to customize how the robot behaves. Our work takes a step in this direction by enabling robots to detect when none of the hypotheses they have explain the user input, and our experiments show promising results. Of course, there are still limitations to this. One limitation is in the experiments themselves, which are only for motion planning tasks with low-dimensional hypothesis spaces. A more fundamental limitation is that there will still be cases when the person wants something outside the robot's hypothesis space, but the robot can nonetheless explain their current input relatively well with what it has access to, thus confusing misspecification for slight noise in the human input. This will especially be the case as the hypothesis space is more expressive, and can only be solved by the robot receiving a lot more human input: each might be explainable by some hypothesis, but eventually no hypothesis can explain all input. More work is needed in studying how to query for diverse human input, as well as how to convey what the robot has learned back to the person, and in general how to have a true collaborative interaction to detect and resolve misspecification in the objective space.

9.1 Related Work

We group prior work into three main categories: enabling robots to learn from human input, doing so while leveraging uncertainty, and estimating model confidence.

9.1.1 Robots learning from humans

The programming of robots through direct human interaction is a well-established paradigm. Human input can be given to the robot in a variety of forms, from teleoperation of the robot by a user to kinesthetic teaching [11].

In such interaction paradigms, the robot aims to infer a *cost function* or *policy* that best describes the examples that it has received. New avenues of research focus on learning such robot objectives from human input through demonstrations [3, 167], teleoperation data [102], corrections [99, 19], comparisons [55], examples of what constitutes a goal [80], or even specified proxy objectives [89]. In this paper, we focus on learning from two of such types of human input – demonstrations and physical corrections – although we stress that the principles outlined in our formalism are more general and could be applied to the other interaction modes mentioned.

One approach to learning behaviors from human inputs is inverse reinforcement learning (IRL). In classical IRL, the robot receives complete optimal *demonstrations* of how to perform a task, and the robot learns the human's cost function from these observations [109, 162, 167]. In this paradigm, it is typically assumed that the expert is trying to optimize an unknown cost function. The robot uses the observations of the human's behavior to recover the underlying objective.

Another useful form of human input are *corrections*: here, the robot performs the task according to how it was programmed and the user corrects aspects of the task to better match their preferences. From these sparse interactions, the robot also performs cost function inference to improve performance during the next task iteration [197, 175, 113]. Examples of learning from corrections have been explored in offline [99],[85] and online settings [19, 18, 43, 12].

Although powerful, the aforementioned IRL works assume that the human expert provides optimal demonstrations, which is often an unrealistic assumption. Real human input, especially during interaction with high degree-of-freedom systems like robotic manipulators, is noisy and sub-optimal. Second, much of the corrections literature has focused on estimates of the human's objectives. However, in practice, even the most likely estimate might not be a very likely one. Thus, in both domains, we stress that it is important to maintain the uncertainty over the estimated objectives.

9.1.2 Uncertainty in robot learning

Rather than estimating a single objective, some learning methods maintain an entire probability distribution over what the objective might be [38, 90, 144, 173]. This not only enables the robot to leverage a prior, but also to then generate its behavior in a way that is mindful of the entire distribution, rather than just using the the maximum likelihood estimator.

Bayesian IRL [173] treats demonstrations as evidence about the objective, and does a Bayesian belief update on a prior distribution. Inverse Reward Desing [90] treats the objective a designer specified for a particular set of environments (a "proxy" objective) as evidence about the true desired objective, again obtaining a full distribution over what the designer might actually want. The intuition is that this observed proxy objective (that may be misspecified) incentivizes behavior that is approximately optimal with respect to the true objective.

Lastly, specifically for input as physical corrections, [144] reasons over the uncertainty of the estimated human preferences through the means of a Kalman filter. The method maintains a mean estimate and a covariance of this estimate as a measure of confidence. These are used in planning the robot's trajectory such that it optimizes for features it is confident about, while avoiding features it is uncertain about.

Although they maintain a full distribution, these works still assume that what the human wants is in the robot's objective space. We argue that this is not necessarily a realistic assumption, and later showcase some consequences that arise when it is not true. When the robot's hypothesis space is misspecified, even when maintaining uncertainty over the objective, state-of-the-art methods interpret human input as evidence about which hypothesis is correct, rather than considering whether any hypothesis is correct. In this work, we focus on the latter.

9.1.3 Situational confidence estimation

Some recent works are studying how to enable robots to understand that their models cannot explain human input well [233, 74, 77]. The authors in [74, 77] employ a noisily-optimal model of human pedestrian motion when the human and the robot operate on separate dynamical systems (and have separate objective functions). The paper introduces the notion of model confidence estimation and uses the apparent likelihood of the human's choice of actions to adjust the confidence in predictions about their behavior.

This work draws inspiration from the notion of model confidence estimation, generalizing it to the setting of inferring what the robot's objective ought to be. Instead of focusing on misspecification of a discrete set of physical goal locations for pedestrian navigation, here we study misspecification of a relatively complex set of possible robot objectives in motion planning tasks. As a result of focusing on robot objectives, we also study a different form of human input – that is, input in the context of operating on the same dynamical system, such as full task demonstrations and physical corrections.

9.2 Problem Formulation and Approach

We consider a robot R operating in the presence of a human H whom it seeks to assist in the execution of some task. In the most general setting, the robot and the human are both able to affect the evolution of the state $x \in \mathbb{R}^n$ over time through their respective control inputs:

$$x^{t+1} = f\left(x^t, u_R^t, u_H^t\right) \ , \tag{9.1}$$

with $u_R \in \mathcal{U}_R$ and $u_H \in \mathcal{U}_H$, where $\mathcal{U}_i$ $(i \in \{H, R\})$ are compact sets. We assume that the human has some consistent preference ordering between different state trajectories and input signals, which could in principle be expressed through a cost function of the form

$$C^*(\mathbf{x}, \mathbf{u}_R, \mathbf{u}_H) \tag{9.2}$$

where the state trajectory is $\mathbf{x} = [x^0, x^1, \dots, x^T] \in \mathbb{R}^{n(T+1)}$, the robot's control input is $\mathbf{u}_R = [u_R^0, u_R^1, \dots, u_R^T] \in \mathbb{R}^{n(T+1)}$, and the human's is $\mathbf{u}_H = [u_H^0, u_H^1, \dots, u_H^T] \in \mathbb{R}^{n(T+1)}$.[1] Note that this hypothesized cost function C^* can be quite general, encoding an arbitrary preference ordering. However, the robot does not in general have access to the human's preferences C^*, and must instead attempt to infer and represent them tractably.

In order to do this, the robot can typically reason over a parametrized approximation of the cost function, which introduces an inductive bias, making inference tractable at the cost of limiting expressiveness: in some cases, the chosen set of parametric functions may fail to encode preferences that would explain the human's behavior with sufficient accuracy. In this work, we will denote by C_θ the cost function induced by parameters $\theta \in \Theta$, and the robot seeks to estimate the human's preferred θ from her control inputs $\mathbf{u}_H$.

In a general setting, since the state trajectory $\mathbf{x}$ is determined not only by the human's actions $\mathbf{u}_H$ but also the robot's $\mathbf{u}_R$, the human would need to reason about how the robot will respond to her decisions. This requires analyzing the interaction in a game-theoretic framework [89, 76], which will not be the object of this work. Instead, we focus on common interaction scenarios in which the robot can approximately assume that the human does not explicitly account for the coupled mutual influence between both agents' decisions. This happens frequently if the human is either providing a demonstration for the robot or intervening to correct the robot's default behavior. In these settings, the typical assumption is that the human has all necessary information about the robot's control input $\mathbf{u}_R$ before deciding on her own $\mathbf{u}_H$.

[1]For deterministic dynamics (9.1), having $x^0, \mathbf{u}_R$ and $\mathbf{u}_H$ is enough to fully specify the entire state trajectory $\mathbf{x}$. In this case, the cost function could be rewritten as $C^*(x^0, \mathbf{u}_R, \mathbf{u}_H)$ by implicitly encoding (9.1). For clarity, we use the more general form in (9.2) and make the dependence explicit where needed.

Thus, given observations of the human input $\mathbf{u}_H$ from an initial state x^0, the robot needs to draw inferences on the cost parameter θ:

$$P(\theta \mid x^0, \mathbf{u}_R, \mathbf{u}_H) = \frac{P(\mathbf{u}_H \mid x^0, \mathbf{u}_R; \theta) P(\theta)}{\int_{\bar{\theta}} P(\mathbf{u}_H \mid x^0, \mathbf{u}_R; \bar{\theta}) P(\bar{\theta}) d\bar{\theta}} \ , \tag{9.3}$$

where $P(\mathbf{u}_H \mid x^0, \mathbf{u}_R; \theta)$ characterizes how the robot expects the human's input to be informed by her preferences, conditioned on the initial state and the robot's expected controls.

For example, if the human were assumed to act optimally, this model would place all probability on the set of optimal states and actions with respect to the cost C_θ. Of course, this would be an unreasonably strong assumption given that the robot's parametrized cost constitutes a best effort to approximate the human's preferences. Instead, a useful modeling choice can be to characterize the human as being more *likely* to take actions that are well-aligned with her preferences.

One such model is inspired by the Boltzmann energy-based model satisfying the maximum entropy principle [103]. Following its adaptations as a model of human decision-making in [217, 21, 19], we model the human as a noisily-optimal agent that tends to choose control inputs that approximately minimize the modeled cost:

$$P(\mathbf{u}_H \mid x^0, \mathbf{u}_R; \theta, \beta) = \frac{e^{-\beta C_\theta \left(\mathbf{x}(\cdot; x^0, \mathbf{u}_R, \mathbf{u}_H), \mathbf{u}_R, \mathbf{u}_H\right)}}{\int_{\bar{\mathbf{u}}_H} e^{-\beta C_\theta \left(\mathbf{x}(\cdot; x^0, \mathbf{u}_R, \bar{\mathbf{u}}_H), \mathbf{u}_R, \bar{\mathbf{u}}_H\right)} d\bar{\mathbf{u}}_H}. \tag{9.4}$$

In this model, the inverse temperature coefficient $\beta \in [0, \infty)$ determines the degree to which the robot expects to observe human actions that are consistent with the cost model.

The goal is to detect when the robot does not have a rich enough hypothesis space, i.e. when C^* lies far outside of any C_θ. We call this problem *objective space misspecification*. Rather than only interpreting human input as evidence about *which* hypothesis is correct, we additionally focus on considering whether *any* hypothesis is correct. It is thus crucial that the robot can quantify the extent to which any parameter value $\theta \in \Theta$ can correctly explain the observed human input.

9.2.1 Situational confidence estimation

The key to our approach goes back to the inverse temperature parameter β in (9.4). Typically, β is a fixed term, encoding the degree to which the robot expects to observe human actions that are optimal. Setting it to 0 models a randomly-acting human, while setting it to ∞ models a perfectly optimal human. However, the possibility of objective space misspecification brings fixing β into question: when the space is correctly specified, we would expect the human actions to indeed be somewhat close to optimal; but when the space is misspecified, *we should expect the actions to be far from optimal for any θ.* Thus, rather than treating β as a fixed term, we build on the work in [74, 77] and explicitly reason over β as an additional inference parameter along with θ. Since β directly impacts the entropy of the human's decision model, it can be used as an effective and computationally efficient measure of the robot's confidence in its parametric interpretation of the human's preference: we say that the robot is assessing its *situational confidence* for the inference task at hand.

Thus, the robot maintains a joint Bayesian belief $b(\theta, \beta)$. For each new measurement of $\mathbf{u}_H$ given $x^0, \mathbf{u}_R$, this belief is updated as:

$$b'(\theta,\beta) = \frac{P(\mathbf{u}_H \mid x^0, \mathbf{u}_R; \theta, \beta) b(\theta,\beta)}{\int_{\bar{\theta},\bar{\beta}} P(\mathbf{u}_H \mid x^0, \mathbf{u}_R; \bar{\theta}, \bar{\beta}) b(\bar{\theta},\bar{\beta}) d\bar{\theta} d\bar{\beta}} \ , \tag{9.5}$$

where $b'(\theta,\beta) = P(\theta, \beta \mid x^0, \mathbf{u}_R, \mathbf{u}_H)$.

This inference can be seen as analogous to performing Bayesian Inverse Reinforcement Learning [173] with the Maximum Entropy Inverse Optimal Control [**maxent**] observation model, where we maintain the full belief instead of just the maximum likelihood estimate, and we explicitly reason over the additional scaling parameter β. By actively performing inference over β, the robot can gain insight into the reliability of its human model in light of new evidence.

Context-dependent usage of situational confidence

How this insight should be used is dependent on the context of the robot's operation. Here, we provide some examples of how situational confidence can be integrated into various human-robot interaction scenarios and robot motion planners.

In collaborative settings where the human and robot are accomplishing a task together (e.g. manipulating an object together), it may be desirable for the robot to stop and ask for clarification from the human whenever sufficient probability mass indicates low confidence:

$$\forall \theta \in \Theta, \arg\max_{\beta} b'(\beta \mid \theta) < \epsilon \ . \tag{9.6}$$

That is, for a predefined threshold ϵ, if all hypotheses have the most mass on βs lower than ϵ, the robot can raise a flag.

In assistive applications, where the robot is carrying out a task in close physical proximity to the human, the robot may receive intermittent human input to correct it's task performance. In such scenarios, it may be appropriate for the robot to simply dismiss human corrections that it cannot explain in terms of modeled preference parameters and carry on with its pre-defined task. That is, when a human input results in a $b'(\theta,\beta)$ that satisfies (9.6), the input gets discarded.

Situational confidence could also be leveraged by robot motion planners that excel at decision making under uncertainty. Here, the robot may use its joint posterior belief

$b'(\theta, \beta)$ to make goal-driven decisions in the presence of the human. To this end, the coupling between the inference problem and the robot's planning problem can be viewed as a partially observable Markov decision process (POMDP), where the hidden parts of the state are the cost parameter θ and the situational confidence β, the robot receives observations about them via human actions $\mathbf{u}_H$, it takes actions $\mathbf{u}_R$, and it optimizes an unknown parametrized cost C_θ. Our problem is, thus, akin to identifying misspecification in the state space of the POMDP. However, inference and planning in such spaces requires solving the full POMDP, which is computationally intractable for large, real-world problems [107].

Alternative, less computationally demanding motion planning approaches are also amenable to our framework, where the robot plans to minimize the expected cost for the human given its current belief, by marginalizing over β:

$$\min_{\mathbf{u}_R} \mathop{\mathbb{E}}_{\theta \sim b} \left[C_\theta(\mathbf{x}, \mathbf{u}_R, \mathbf{u}_H) \right] \ , \tag{9.7}$$

for an expected human input $\mathbf{u}_H$ that will typically be $\mathbf{0}$ if the robot is attempting to successfully perform the task without the need for active human intervention. To understand the implication (9.7) has as a function of the inference over β, we need to understand the posterior belief marginalized over β that we are taking the expectation over. At one extreme, if for all θs the conditional distribution $b'(\beta \mid \theta)$ puts all probability mass on $\beta = 0$ (i.e. input poorly explained), since $P(\mathbf{u}_H \mid x^0, \mathbf{u}_R; \theta, \beta = 0)$ is the same for all θs, the robot will obtain a posterior for θ that is equal to the prior. The optimization above becomes the same as optimizing using the robot's prior, i.e. the robot ignores the human input. At the other extreme, if there is one θ that perfectly explains the input and all others do not, the posterior will put all probability mass on that θ, and the robot will switch to optimizing it.

The objective expectation may also be appropriately weighted by the robot's situational confidence for each θ:

$$\min_{\mathbf{u}_R} \mathop{\mathbb{E}}_{\theta, \beta \sim b} \left[\beta C_\theta(\mathbf{x}, \mathbf{u}_R, \mathbf{u}_H) \right] \ , \tag{9.8}$$

which leads to the robot prioritizing those components of the task about which it is most certain.

In Sections 9.3 and 9.4 we discuss some of these possibilities in the context of learning from demonstrations and corrections.

9.2.2 Cost representation through basis functions

One way to approximate the infinite-dimensional space of possible cost functions using a finite number of parameters is the use of a finite family of basis functions Φ_i[162]. This family can be seen as a truncation of an infinite collection of basis functions spanning the full function space. Parametric approximations C_θ of the cost function C^* then have the form

$$C_\theta(\mathbf{x}, \mathbf{u}_R, \mathbf{u}_H) = \sum_{i=1}^{d} \theta^i \Phi_i(\mathbf{x}, \mathbf{u}_R, \mathbf{u}_H) = \theta^T \Phi(\mathbf{x}, \mathbf{u}_R, \mathbf{u}_H) \ . \tag{9.9}$$

Consistent with classical utility theories [217], we further assume that the human's preferences can be approximated through a cumulative return over time, rewriting (9.9) as

$$C_\theta(\mathbf{x}, \mathbf{u}_R, \mathbf{u}_H) = \sum_{i=1}^{d} \theta^i \sum_{t=0}^{T} \phi_i(x^t, u_R^t, u_H^t) \ , \tag{9.10}$$

where $\phi_i : \mathbb{R}^n \times \mathcal{U} \times \mathcal{U} \to \mathbb{R}$ are fixed, pre-specified, bounded real-valued basis functions, θ is the unknown parameter that the robot is trying to fit according to the human's preferences, and d is the dimensionality of its domain Θ.

In the domains presented in Sections 9.3 and 9.4, the functions ϕ_i output feature values that encode key aspects of a task—for example distance between the robot body and obstacles in the environment, speed of the motion, or characteristics of a motion planning task. In general, the ϕ_i can either be hand-engineered by a system designer or more generally learned through data-driven approaches [70].

It is important to stress that the misspecification issue we are trying to mitigate is quite general and does not exclusively affect objectives based on hand-crafted features: any model could ultimately fail to capture the underlying motivation of some human actions. While it may certainly be possible, and desirable, to continually increase the complexity of the robot's model to capture a richer space of objectives, there will still be a need to account for the presence of yet-unlearned components of the true objective. In this sense, our work is complementary to open-world objective modeling efforts.

Note that, using a cost model in the form of (9.10), the observation model (9.4) becomes overparametrized, since for any (θ, β) pair with $\theta \in \Theta$ and $\beta \in [0, \infty)$, one can always find a different $\theta' = c\theta$ with an associated $\beta' = \beta/c$ leading to the same probability distribution over human choices. This is equivalent to using an unrestricted Θ and $\beta = \|\theta\|$. Due to this overparametrization, the absolute value of β does not have a universal meaning, and restricting θ to have a fixed norm is necessary in order to make comparisons between the β values associated to different θ hypotheses. We thus restrict our Θ to the set of vectors with unit norm.

Consider the case where the human provides input for a cost function in the robot's objective space. This results in the robot inferring high probability on the corresponding θ vector on the unit sphere with a high magnitude β. However, if the cost that the human cares about and provides input for is outside the robot's hypothesis space, the robot will infer low probability on all θ vectors in the unit sphere, with low magnitude βs.

We now proceed by describing the explicit algorithmic approaches to inferring situational confidence in the learning from demonstrations and corrections domains.

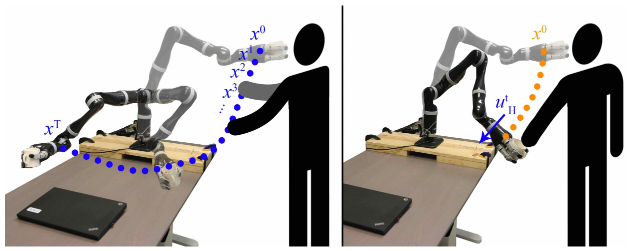


Figure 9.2: (Left) Visual example of a full human-provided demonstration $\mathbf{x}$. (Right) Visual example of a human physical correction u_H^t onto the robot's current trajectory $\mathbf{x}$.

9.3 Algorithmic Approach: Demonstrations

9.3.1 Formulation

In learning from demonstrations, the human directly controls the state trajectory $\mathbf{x}$ through her input $\mathbf{u}_H$, which enables her to offer the robot a demonstration of how to perform the task. Fig. 9.2 (left) is an example of such a demonstration.

During the demonstration, the robot is often put in gravity compensation mode or is teleoperated, to grant the person full control over the desired trajectory. As such, in this setting, the cost function C_θ does not depend on the robot controls $\mathbf{u}_R$. Additionally, since the person is primarily concerned with the robot's states and not with the (robot or human) actions required to reach those states, we model the human's internal preferences as only dependant on the state trajectory $\mathbf{x}$. Accordingly, the cost function in (9.10) becomes:

$$C_\theta(\mathbf{x}) = \theta^T \Phi(\mathbf{x}). \tag{9.11}$$

The cost does not have a direct dependence on the actions, but it has an indirect one, as $\mathbf{x}$ depends on $\mathbf{u}_R$ and $\mathbf{u}_H$.

In our problem formulation, we would like the robot to explicitly reason about how well it can explain the demonstration given its human model. Thus, we can adapt the model in (9.4) to use this new cost function[2],

$$P(\mathbf{x} \mid \theta, \beta) = \frac{e^{-\beta\theta^T\Phi(\mathbf{x})}}{\int_{\bar{\mathbf{x}}} e^{-\beta\theta^T\Phi(\bar{\mathbf{x}})} d\bar{\mathbf{x}}} \ , \tag{9.12}$$

then perform the Bayesian update in (9.5)

$$b'(\theta, \beta) = \frac{P(\mathbf{x} \mid \theta, \beta) b(\theta, \beta)}{\int_{\bar{\theta},\bar{\beta}} P(\mathbf{x} \mid \bar{\theta}, \bar{\beta}) b(\bar{\theta}, \bar{\beta}) d\bar{\theta} d\bar{\beta}} \ . \tag{9.13}$$

Given $b'(\theta, \beta)$, we now can use any of (9.6), (9.7) or (9.8). Next, we discuss making inference with (9.12) and (9.13) tractable.

[2]For deterministic (9.1), $P(\mathbf{u}_H \mid x^0, \mathbf{u}_R; \theta, \beta)$ is equivalent to $P(\mathbf{x} \mid \theta, \beta)$.

9.3.2 Approximation

Although the proposed formalism enables us to capture if the robot's hypothesis space cannot explain the human's input, it is non-trivial to implement tractably for continuous β and θ, and large state and action spaces. Concretely, notice that equations and (9.12) and (9.13) constitute a doubly-intractable system with denominators that cannot be computed exactly. For this reason, we employ several approximations in order to demonstrate the benefits of estimating situational confidence. Note that we do not consider these a contribution of our work: we choose the simplest approximations that facilitate tractability. There are many methods for approximate inference of θ studied in the literature that could be used for the joint (θ, β) spaces as well, from Metropolis Hastings [90, 184], to acquiring an MLE only via importance sampling of the partition function [70] or via a Laplace approximation [135].

To approximate the intractable integral in (9.12), we sampled a set $\mathcal{X}$ of 1500 trajectories. We sampled costs according to (9.11) given by random unit norm θs, then optimized them with an off-the-shelf trajectory optimizer. We used TrajOpt [192], which is based on sequential quadratic programming and uses convex-convex collision checking. This way, we obtain dynamically feasible trajectories that optimize for different features in varying proportions. While this sampling strategy cannot be justified theoretically, it works well in practice: the resulting optimized trajectories are a heuristic for sampling diverse and interesting trajectories in the environment. Future work will address this shortcoming by either providing theoretical guarantees or using importance sampling instead.

For the second approximation to (9.13), we discretized the space of $\theta \in \Theta$ and $\beta \in \mathcal{B}$ into sets Θ_D and $\mathcal{B}_D$, which leaves us with a finite, easy to compute posterior. For more practical details on specific discretization schemes, see Appendix 9.8.1.

Using the above discretization[3], we can now perform tractable inference from demonstrations $\mathcal{D}$ to obtain a discrete posterior $b(\theta, \beta)$. Algorithm 2 summarizes the full procedure: given $\Theta_D, \mathcal{B}_D, \mathcal{X}$, and $\mathcal{D}$, our method iteratively updates the belief using (9.12) and (9.13), resulting in the posterior $b(\theta, \beta)$. Lacking any a-priori information, we chose a uniform prior but our method will work with any prior. We next present examples for what this posterior looks like in different scenarios.

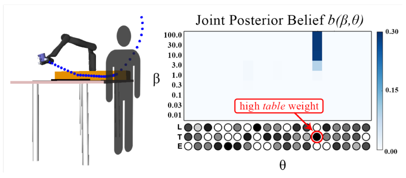
9.3.3 Examples

To provide intuition for how situational confidence can indicate when a robot's hypothesis space is misspecified, we illustrate some examples with a robot manipulator learning

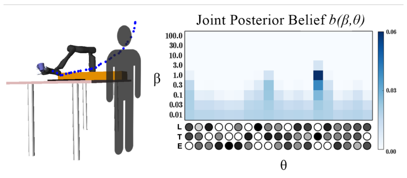
[3]In situations where the designer might want high fidelity inference over a large space of θ vectors, reasoning over a heavily discretized space would be more computationally expensive. However, longer offline computation is possible in our learning-from-demonstrations scenario as the inference happens offline, after providing the robot with human demonstrations. Alternatively, we could use Monte Carlo sampling approaches, similar to [90, 173].

Algorithm 2 Learning from Demonstrations (Offline)

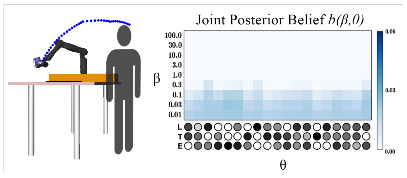
```
Require: Discretized sets $\Theta_D, \mathcal{B}_D, \mathcal{X}$, set of demonstrations $\mathcal{D}$.
Ensure: Posterior belief $b(\theta, \beta)$ inferred from $\mathcal{D}$.
  $b(\theta, \beta) \leftarrow Uniform(\theta, \beta)$.
  for x in $\mathcal{D}$ do
      for all $\theta \in \Theta_D, \beta \in \mathcal{B}_D$ do
          $P(\mathbf{x} \mid \theta, \beta) = \frac{e^{-\beta\theta^T\Phi(\mathbf{x})}}{\sum_{\bar{\mathbf{x}}\in\mathcal{X}} e^{-\beta\theta^T\Phi(\bar{\mathbf{x}})}}$ as per (9.12).
          $b(\theta, \beta) \leftarrow \frac{P(\mathbf{x}|\theta,\beta)b(\theta,\beta)}{\sum_{\bar{\theta}\in\Theta,\bar{\beta}\in\mathcal{B}} P(\mathbf{x}|\bar{\theta},\bar{\beta})b(\bar{\theta},\bar{\beta})}$ as per (9.13).
      end for
  end for
```



(a) (Left) Simulated perfect demonstration with the objective to keep the cup close to the table. (Right) Posterior belief resulted from this demonstration. Notice that a perfect demonstration leads to a high probability on the correct θ and high values for β.



(b) (Left) Noisy human demonstration with the objective to keep the cup close to the table. (Right) Posterior belief resulted from this demonstration. Notice that a noisy but well-explained demonstration leads to a high probability on the correct θ and moderately high values for β. However, the noise in the demonstration significantly reduces the probability at the distributional peak.

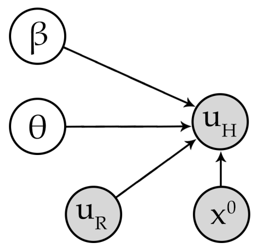


(c) (Left) Simulated perfect demonstration with the objective to keep the cup away from the human's body. (Right) Posterior belief resulted from this demonstration. Notice that, since this demonstration is poorly explained (the robot is not reasoning about distance from the human), the posterior belief is spread out approximately uniformly over all θs and the lowest β values. This indicates that the robot cannot tell what the demonstration was intended for.

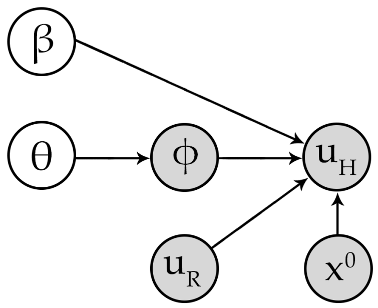
Figure 9.3: Three examples of demonstrations and the inferred posterior belief after each one of them. The robot infers the right $\theta = [0, 1, 0]$ from the two well-explained demonstrations, but, unlike the perfect simulated demonstration in 9.3(a), the noisy one in 9.3(b) cannot reach the highest β and has as overall more spread-out probability distribution with a lower peak value. Lastly, the perfect simulated demonstration that is poorly explained in 9.3(c) results in a posterior that is spread-out over all θs and the lowest βs , consistent with the robot not being able to tell what the human's objective was.

from a human demonstrator. These examples help prepare the setup we will present in our actual experiments in Section 9.5.

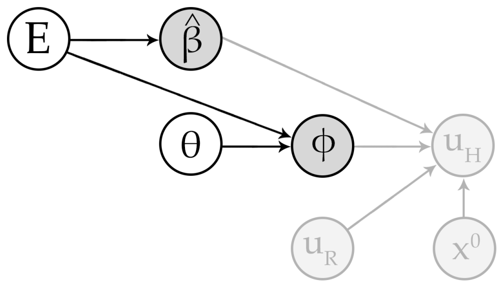
The robot manipulator is performing a household task of moving cups from a shelf



(a) In the true graphical model, u_H is an observation of θ and the situational confidence β.



(b) We use the proxy variable Φ to first estimate β efficiently.



(c) We interpret the estimate $\hat{\beta}$ as an indirect observation of the unobserved E, which we then use for the θ estimate.

Figure 9.4: Graphical model formulation (a) and modifications to it ((b) and (c)) for real-time tractability.

onto the kitchen table. The robot needs to learn from the person's demonstrations how to best perform this task. For this purpose, the person physically guides the robot through one or a few demonstrations of moving the cup down to the table, from which the robot infers the hidden objective function.

In these examples, the robot's hypothesis space includes three features: efficiency (E) as sum of squared velocities over the trajectory, keeping the cup close to the table (T), and keeping the cup away from the laptop (L) depicted in black.

Formally, we can represent these three feature mappings as:

$$\Phi(\mathbf{x}) = \begin{bmatrix} \sum_{i=1}^{T}((x^i - x^{i-1})/\Delta t)^2 \\ \sum_{i=0}^{T} ||x^i - x_{\text{table}}||_2 \\ \sum_{i=0}^{T} \max\{0, L - ||x^i - x_{\text{laptop}}||_2\} \end{bmatrix} \tag{9.14}$$

where L is the radius of a penalty sphere around the laptop, Δt is the discrete timestep between the states in the trajectory, and the corresponding feature weight vector is $\theta \in \mathbb{R}^3$.

Fig. 9.3 demonstrates how the feature weight θ and the situational confidence β are affected for well-explained, noisy, and poorly-explained simulated human demonstration. The posterior belief is shown for the combination of discrete parameters θ and β. Higher β values indicate higher situational confidence. The three circles under each column represent the θ vector for that column, with the components being the efficiency, distance from the table, and distance from the laptop features. A larger feature weight is indicated by a darker colored circle, while a white color indicates zero weight.

First, in 9.3(a), we consider the case where the demonstration is a perfectly optimal trajectory produced by TrajOpt [192]. This serves as a sanity check for when the human and the robot have the same hypothesis space and the demonstration is perfect. The optimal demonstration was produced by finding a trajectory that moves the cup from the start configuration to the end while minimizing the distance between the cup and the table. Notice that, with a perfect demonstration, the posterior distribution places the most probability mass on the θ that indicates high penalties for staying away from the table but no penalties for lack of efficiency or closeness to laptop. Moreover, the posterior also reveals that the most likely θ also corresponds *with the highest available confidence* β.

Next, in 9.3(b) we recorded a real human demonstration of the same cup-to-table behavior. The nature of demonstrations both on hardware and from real people introduce noise into the demonstration, making it potentially suboptimal with respect to the robot's model. However, in this case the human and the robot still share the same hypothesis space (i.e. the robot and the human both know about the the efficiency, table, and laptop features). Here, we study how the noise in the demonstration affects the robot's inference. Notice that even with an imperfect demonstration, the robot is able to identify the correct θ parameter, but now with a lower confidence β.

Lastly, we consider the example where the demonstration is optimal but the robot does not have a rich enough hypothesis space to explain it. The robot reasons about the same three features, but now the demonstration was produced by optimizing for an additional feature that is outside its hypothesis space: keeping the cup away from the human's body. We observe that the probability distribution in 9.3(c) is spread over all the θ values in the space, with the highest values on low βs. This example shows how, in the case of poorly-explained input, the robot's inference is unsure which objective the human had in mind, and assigns low situational confidence to the given input.

These illustrative examples give us valuable insight into how the (θ, β)-belief changes depending on how well-explained the input is. For perfectly explained demonstrations, the inference identifies the correct θ with high posterior probability. As the input becomes more poorly-explained, the robot loses confidence in all θs, assigning approximately uniformly spread-out probability on the lowest situational confidence values β.

9.4 Algorithmic Approach: Corrections

9.4.1 Formulation

We consider the setting in which human input is provided in the form of physical interventions during the robot's task execution. Fig. 9.2 (right) is an example of such a correction. The human may provide a correction to improve some aspect of the task execution that is not represented in the robot's objective space. When the robot receives input, it should be able to reason about its situational confidence in light of the correction and replan its trajectory accordingly for the rest of the task execution or until a new correction happens. Thus, the robot must have access to an inference algorithm that can run in real time. In this section, we will present an online version of our situational confidence framework.

In the physical corrections setting, the robot starts with an initial guess of the parameter θ and uses a trajectory optimization scheme to compute a motion plan seeking to minimize the associated cost C_θ. The robot performs the task at hand by applying controls $\mathbf{u}_R$ via an impedance controller in order to track the computed trajectory $\mathbf{x}$.

At any timestep t during the trajectory execution, the human may physically interact with the robot, inducing a joint torque u_H^t. When this happens, the robot can use the human input to update its estimated θ parameter, and thereby the corresponding objective C_θ. Given the new adapted objective, the robot replans an optimized trajectory $\mathbf{x}$ and tracks it until the next human input is sensed or until the task is completed.

Following [19], the robot's representation of the task assumes that the human does not explicitly care about the robot's control effort, but only about features of the state trajectory. In addition, the human is assumed to have a preference for minimizing her own control effort. This captures the human's incentive to have the robot perform the task autonomously, providing only minimal input to guide the robot towards the correct behavior when necessary. Encompassing these assumptions, the cost (9.10) takes the form:

$$C_\theta(\mathbf{x}, u_H^t) = \theta^T\Phi(\mathbf{x}) + \lambda\|u_H^t\|^2. \tag{9.15}$$

To approximately compute the trajectory resulting from the human's input, we follow the approach in [19] and introduce the notion of a *deformed trajectory* $\mathbf{x}_D$. This trajectory constitutes the robot's estimate of the human's desired trajectory given her applied torque u_H^t. Given the robot's default trajectory $\mathbf{x}_R := \mathbf{x}(\cdot; x^0, \mathbf{u}_R, \mathbf{0})$ and having observed the instantaneous human intervention u_H^t, we compute $\mathbf{x}_D$ by deforming the robot's default trajectory in the direction of u_H^t:

$$\mathbf{x}_D = \mathbf{x}_R + \mu A^{-1}\tilde{\mathbf{u}}_H \ , \tag{9.16}$$

where $\mu > 0$ scales the magnitude of the deformation, $A \in \mathbb{R}^{n(T+1)\times n(T+1)}$ defines a norm on the Hilbert space of trajectories[4] and dictates the deformation shape [66], and $\tilde{\mathbf{u}}_H \in \mathbb{R}^{n(T+1)}$ is u_H^t at indices nt through $n(t+1)$ and 0 otherwise. The human is therefore modeled by (9.15) as trading off between inducing a good trajectory $\mathbf{x}_D$ with respect to θ, and minimizing her effort.

Equipped with this cost function, we need the robot to reason about the reliability of its objective space given new inputs in the form of corrections. In contrast with our analysis in Section 9.3, here the person does not give full demonstrations $\mathbf{x}$, but instead offers corrections u_H^t based on the robot's default trajectory $\mathbf{x}_R$. Applying (9.4) to this setting, we have:

$$P(u_H^t \mid x^0, \mathbf{u}_R; \theta, \beta) = \frac{e^{-\beta(\theta^\top\Phi(\mathbf{x}_D)+\lambda\|u_H^t\|^2)}}{\int e^{-\beta(\theta^\top\Phi(\bar{\mathbf{x}}_D)+\lambda\|\bar{u}\|^2)}d\bar{u}} \ , \tag{9.17}$$

[4]We used a norm A based on acceleration, consistent with [19], but other norm choices are possible as well.

where $\mathbf{x}_D$ and $\bar{\mathbf{x}}_D$ are given by (9.16) applied to their respective controls u_H^t and $\bar{u}$.

Ideally, with this model of human actions, illustrated in Fig. 9.4(a), we would perform inference over both the situational confidence β and the modeled parameters θ by maintaining a joint Bayesian belief $b'(\theta, \beta)$. Analogously to the demonstrations case, our probability distribution over θ would automatically adjust for well-explained corrections, whereas for poorly-explained ones the robot's posterior would not deviate significantly form its prior on θ. Unfortunately, this Bayesian update is not generally feasible in real time, given the continuous and possibly high-dimensional nature of the parameter space Θ. Even in simple scenarios with a small number of continuous features, discretizing Θ as we did in the demonstrations case would generally yield an overly slow inference, making the method impractical for use in the real-time collaborative scenarios that we are interested in here. Thus, to evaluate the benefits of estimating β we need to derive an online method that goes beyond simple discretization.

9.4.2 Approximation

To alleviate the computational challenge of performing joint inference over β and θ, we introduce a structural assumption that will enable us to approximately decouple the two inference problems.

Estimating β

To estimate β without dependence on θ, we will assume that in order to decide what correction to provide, the human will first choose the desired features Φ of the resulting trajectory $\mathbf{x}_D$ and then select an input u_H^t that will obtain these features (Fig. 9.4(b)).

Based on the observed human input u_H^t and the trajectory features of the deformed trajectory $\Phi(\mathbf{x}_D)$, the robot can obtain an estimate of β by considering how efficient the human's input was for the features achieved. Letting $\mathcal{U}_\Phi$ be the set of inputs that achieve the same observed features $\Phi_D := \Phi(\mathbf{x}_D)$, the Boltzmann decision model gives

$$\begin{aligned} P(u_H^t \mid x^0, \mathbf{u}_R; \Phi_D, \beta) &= \frac{e^{-\beta(\theta^\top \Phi_D + \lambda \|u_H^t\|^2)}}{\int_{\mathcal{U}_\Phi} e^{-\beta(\theta^\top \Phi(\bar{\mathbf{x}}_D) + \lambda \|\bar{u}\|^2)} d\bar{u}} \\ &= \frac{e^{-\beta\lambda \|u_H^t\|^2}}{\int_{\mathcal{U}_\Phi} e^{-\beta\lambda\|\bar{u}\|^2} d\bar{u}} \ , \end{aligned} \tag{9.18}$$

since the term $\theta^\top \Phi(\bar{\mathbf{x}}_D)$ is constant for all $\bar{u} \in \mathcal{U}_\Phi$ and equal to the term $\theta^\top \Phi_D$ in the numerator.

Using (9.18), the robot can obtain an estimate of β by considering how efficient the human's correction was for the features achieved—if the input seems highly inefficient, this is indicative that the features modeled by the robot may not accurately capture the human's preference.

It is useful to approximate the integral over the constrained set $\mathcal{U}_\Phi \subset \mathcal{U}$ by an integral over the entire set of possible inputs $\mathcal{U}$, introducing a penalty term in the exponent that results in a soft indicator function for $\bar{u} \in \mathcal{U}_\Phi$:

$$P(u_H^t \mid x^0, \mathbf{u}_R; \Phi_D, \beta) \approx \frac{e^{-\beta\lambda\|u_H^t\|^2}}{\int_{\mathcal{U}} e^{-\beta(\lambda\|\bar{u}\|^2 + \kappa\|\Phi(\bar{\mathbf{x}}_D) - \Phi_D\|^2)} d\bar{u}} \quad . \tag{9.19}$$

Note that for an arbitrarily large κ there is an arbitrarily small probability assigned to $\mathcal{U} \setminus \mathcal{U}_\Phi$ in the integral. It is now possible to apply the Laplace approximation to the unconstrained integral (see Sec. 9.9 for details), yielding:

$$P(u_H^t \mid x^0, \mathbf{u}_R; \Phi_D, \beta) \approx \frac{e^{-\beta\lambda\|u_H^t\|^2}}{e^{-\beta(\lambda\|u_H^*\|^2 + \kappa\|\Phi(\mathbf{x}_D^*) - \Phi_D\|^2)}} \sqrt{\frac{\beta^k |H_{u_H^*}|}{2\pi^k}} \quad , \tag{9.20}$$

where k is the action space dimensionality and $H_{u_H^*}$ is the Hessian of the exponent in the denominator of (9.19) around u_H^*. We obtain the optimal action u_H^* by solving the constrained optimization problem (see Sec. 9.8.2):

$$\begin{aligned} \underset{\tilde{u}_H}{\text{minimize}} \quad & \|\tilde{u}_H\|^2 \\ \text{subject to} \quad & \Phi(\mathbf{x} + \mu A^{-1}\tilde{\mathbf{u}}_H) - \Phi_D = 0 \quad . \end{aligned} \tag{9.21}$$

In other words, the resulting u_H^* is the minimal norm $\tilde{u}_H$ the human could have taken, constrained to lie in $\mathcal{U}_\Phi$. As such, the second norm in the denominator's exponent is 0, and the final conditional probability becomes:

$$P(u_H^t \mid x^0, \mathbf{u}_R; \Phi_D, \beta) = e^{-\beta\lambda(\|u_H^t\|^2 - \|u_H^*\|^2)} \sqrt{\frac{\beta^k |H_{u_H^*}|}{2\pi^k}} \quad . \tag{9.22}$$

We derive below the maximum likelihood estimator (MLE), noting that a maximum *a posteriori* (MAP) estimator is often appropriate given a certain prior on β.

$$\begin{aligned} \hat{\beta} &= \arg\max_{\beta} \{\log(P(u_H^t \mid x^0, \mathbf{u}_R; \Phi_D, \beta)\} \\ &= \arg\max_{\beta} \{-\beta\lambda(\|u_H^t\|^2 - \|u_H^*\|^2) + \log(\sqrt{\frac{\beta^k |H_{u_H^*}|}{2\pi^k}})\}. \end{aligned} \tag{9.23}$$

Applying the first-order condition and setting the derivative to zero yields the maximizer:

$$\hat{\beta} = \frac{k}{2\lambda(\|u_H^t\|^2 - \|u_H^*\|^2)} \quad . \tag{9.24}$$

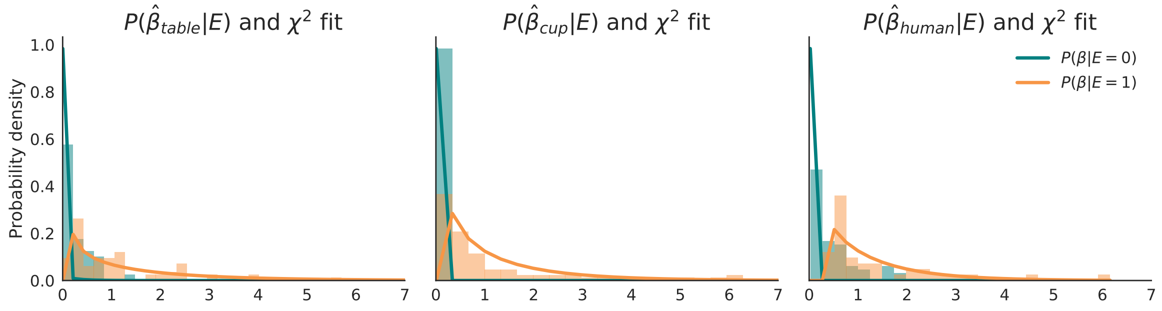


Figure 9.5: Empirical estimates for $P(\hat{\beta} \mid E)$ and their corresponding chi-squared (χ^2) fits.

The estimator[5] above yields a high value when the difference between u_H^t and u_H^* is small, i.e. the person's correction achieves the induced features $\Phi(\mathbf{x}_D)$ efficiently. For instance, if $\mathbf{x}_D$ brings the robot closer to the table, and u_H^t pushes the robot straight towards the table, u_H^t is an efficient way to induce those new feature values. However, when there is a much more efficient alternative (e.g. when the person pushes mostly sideways rather than straight towards the table), $\hat{\beta}$ will be small. Efficient ways to induce the feature values will suggest well-explained inputs, inefficient ones will suggest poorly-explained corrections.

Estimating θ

To tractably estimate θ building on the β estimate, we introduce an auxiliary binary variable $E \in \{0, 1\}$ indicating whether the human's intervention can be well *explained* by the robot's modeled cost features. We will perform offline training with ground-truth access to this variable in order to learn its relation to the robot's estimate $\hat{\beta}$.

When $E = 1$, the human's desired modification of the robot's behavior can be well explained by *some* vector $\theta \in \Theta$, which will lead the intervention to appear less noisy to the robot (i.e. β is large). As a result, the correction u_H^t is likely to be efficient for the cost encoded by this θ. Conversely, when $E = 0$, the intervention appears noisy (i.e. β is small), and the human's correction cannot be well explained by any of the cost features modeled by the robot.

The graphical model depicted in Fig. 9.4(c) relates the induced feature values Φ_D to θ as a function of the E. When $E = 1$, the induced features will tend to have low cost with respect to θ; when $E = 0$, the induced features *do not depend on* θ, and we model them as Gaussian noise centered around the feature values of the robot's currently planned

[5]Note that $\hat{\beta}$ is non-negative, since u_H^* is the minimal-norm $\tilde{u}_H$ that satisfies the constraint, so the difference in the denominator of (9.24) is positive.

trajectory $\mathbf{x}_R$.

$$P(\Phi_D \mid \theta, E) = \begin{cases} \dfrac{e^{-\theta^\top \Phi_D}}{\int e^{-\theta^\top \Phi(\tilde{\mathbf{x}}_D)} d\tilde{\mathbf{x}}_D}, & E = 1 \\ \left(\frac{\nu}{\pi}\right)^{\frac{k}{2}} e^{-\nu ||\Phi_D - \Phi(\mathbf{x}_R)||^2}, & E = 0 \end{cases} \tag{9.25}$$

with the constant in the $E = 0$ case corresponding to the normalization term of the normal distribution.

In addition, this graphical model relates the $\hat{\beta}$ resulting from the model in Fig. 9.4(b) to E by a $P(\hat{\beta} \mid E)$. We fit this distribution from controlled user interaction samples where we have ground-truth knowledge of E[6]. For each sample interaction, we compute $\hat{\beta}$ (for example, using (9.24) if using MLE) and label it with the corresponding binary E value. We fit a chi-squared distribution to these samples to obtain the probability distributions for $P(\hat{\beta} \mid E = 0)$ and $P(\hat{\beta} \mid E = 1)$. The resulting distributions are shown in Fig. 9.5[7].

Using the model in Fig. 9.4(c) with the learned distribution $P(\hat{\beta} \mid E)$, we can infer a θ estimate in real time whenever a physical correction from the human is measured. We do this tractably by interpreting the estimate $\hat{\beta}$ obtained from (9.24) as an indirect observation of the unknown variable E. We combine the empirically characterized likelihood model $P(\hat{\beta} \mid E)$ with an initial uniform prior $P(E)$ to maintain a Bayesian posterior on E based on the evidence $\hat{\beta}$ constructed from human observations at deployment time, $P(E \mid \hat{\beta}) \propto P(\hat{\beta} \mid E)P(E)$.

Further, since we wish to obtain a posterior estimate of the human's objective θ, we use the model from Fig. 9.4(c) to obtain the posterior probability measure

$$P(\theta \mid \Phi_D, \hat{\beta}) \propto \sum_{E \in \{0,1\}} P\big(\Phi_D \mid \theta, E\big) P(E \mid \hat{\beta}) P(\theta) \ . \tag{9.26}$$

Following [19], we note that we can approximate the partition function in the human's policy (9.25) by employing the Laplace approximation. Taking a second-order Taylor series expansion of the exponent's objective about $\mathbf{x}_R$, the robot's current best guess at the optimal trajectory, we obtain a Gaussian integral that can be evaluated in closed form

$$P(\Phi_D \mid \theta, E = 1) \approx e^{-\theta^\top \left(\Phi_D - \Phi(\mathbf{x}_R)\right)} \ . \tag{9.27}$$

We also consider a Gaussian prior distribution of θ around the robot's current estimate $\hat{\theta}$:

$$P(\theta) = \frac{1}{(2\pi\alpha)^{\frac{k}{2}}} e^{-\frac{1}{2\alpha}||\theta - \hat{\theta}||^2} \ , \tag{9.28}$$

[6]Since we tell users what to optimize for, we know whether the human's input is well-explained with respect to the robot's hypothesis space or not.

[7]Because users tend to accidentally correct more than one feature, we perform β-inference separately for each feature. This requires more overall computation (although still linear in the number of features, and can be parallelized) and a separate $P(\hat{\beta} \mid E)$ estimate for each feature.

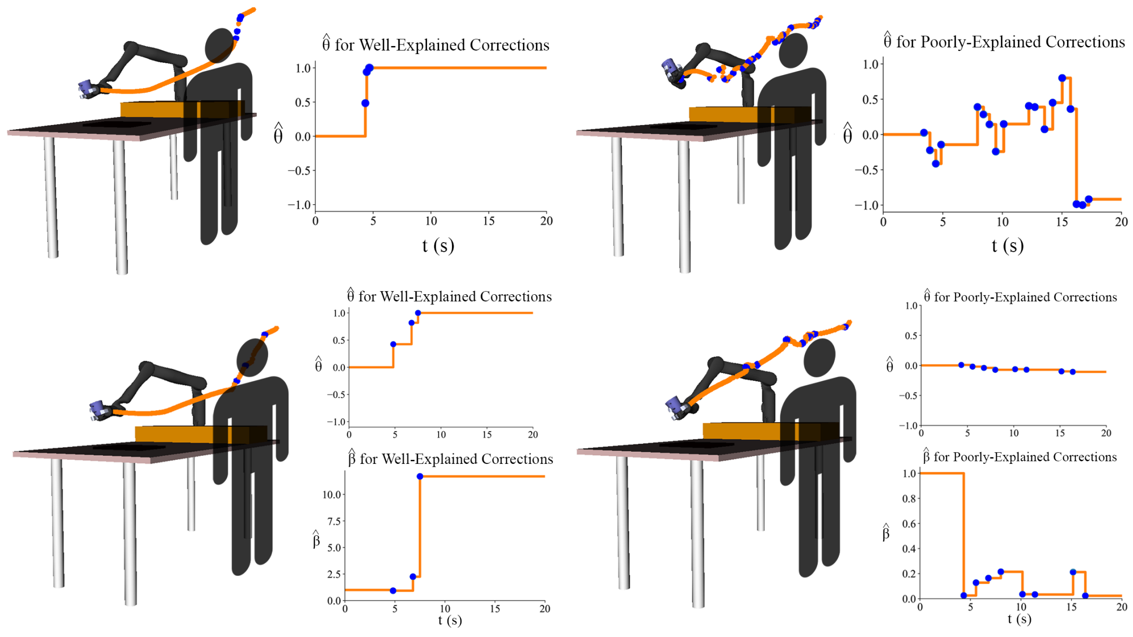


Figure 9.6: Examples of physical corrections (interaction points shown in blue) and the resulting behavior for the fixed β method (top) and estimated β method (bottom). When the corrections are well explained, both methods learn the correct weight $\hat{\theta} = 1.0$. In the case of poorly-explained corrections, our method infers low $\hat{\beta}$ and manages to reduce unintended learning, whereas the fixed β method produces incorrect oscillatory behavior.

where $\alpha \geq 0$ determines the variance of the Gaussian.

To obtain an update rule for the θ parameter, we can simply plug (9.25), (9.27), and (9.28) into (9.26). For legibility, let's denote $\Gamma(\Phi_D, E = i) = P(E = i \mid \hat{\beta})P\big(\Phi_D \mid \theta, E = i\big)$, for $i \in \{0, 1\}$. Then, the maximum-a-posteriori estimate of the human's objective θ is the solution maximizer of

$$\begin{aligned} P(\theta)\Big[\Gamma(\Phi_D, E = 1) + \Gamma(\Phi_D, E = 0)\Big] \\ = \frac{1}{(2\pi\alpha)^{\frac{k}{2}}} e^{-\frac{1}{2\alpha}\|\theta-\hat{\theta}\|^2} \Big[P(E = 1 \mid \hat{\beta}) e^{-\theta^\top\left(\Phi_D - \Phi(\mathbf{x}_R)\right)} \\ + P(E = 0 \mid \hat{\beta}) \left(\frac{\nu}{\pi}\right)^{\frac{k}{2}} e^{-\nu\|\Phi_D - \Phi(\mathbf{x}_R)\|^2}\Big] \ . \end{aligned} \tag{9.29}$$

Differentiating (9.29) with respect to θ and equating to 0 gives the maximum-a-posteriori update rule

$$\hat{\theta}' = \hat{\theta} - \alpha \frac{\Gamma(\Phi_D, E = 1)}{\Gamma(\Phi_D, E = 1) + \Gamma(\Phi_D, E = 0)} \big(\Phi_D - \Phi(\mathbf{x}_R)\big) \ . \tag{9.30}$$

We note that due to the coupling in $\hat{\theta}'$, the solution to (9.30) is non-analytic and can instead be obtained via numerical approaches like Newton-Raphson or quasi-Newton methods.

In previous objective-learning approaches including [19] and [237], it is implicitly assumed that all human actions are fully explainable by the robot's representation of the objective function space ($E = 1$), leading to the simplified update

$$\hat{\theta}' = \hat{\theta} - \alpha\big(\Phi_D - \Phi(\mathbf{x}_R)\big) \quad , \tag{9.31}$$

which can be easily seen to be a special case of (9.30) when $P(E = 0 \mid \hat{\beta}) \equiv 0$. Our proposed update rule therefore *generalizes* commonly-used objective-learning formulations to cases where the human's underlying objective function is not fully captured by the robot's model. We expect that this extended formulation will enable learning that is more robust to misspecified or incomplete human objective parameterizations.[8] Once we obtain the $\hat{\theta}'$ update, we replan the robot trajectory in its 7-DOF configuration space with an off-the-shelf trajectory optimizer, TrajOpt [192].

The update rule changes the weights in the objective in the direction of the feature difference as well, but how much it does so depends on the probability assigned to the correction being well-explained. Looking back at Section 9.2, this update is approximating (9.7). At one extreme, if we know with full certainty that the correction is well explained, then we do the full update as in traditional objective learning. But crucially, at the other extreme, if we know that the correction is poorly explained, we do not update at all and keep our prior belief.

Algorithm 3 Learning from Corrections (Online)

```
Require: $P(\hat{\beta} \mid E = i), \forall i \in \{0, 1\}$ from training data.
  Initialize $\mathbf{x}_R \leftarrow TrajOpt(\hat{\theta})$ for initial $\hat{\theta}$.
  while goal not reached do
      if $u_H \neq \mathbf{0}$ then
          $\mathbf{x}_D = \mathbf{x}_R + \mu A^{-1}\tilde{\mathbf{u}}_H$ .
          $u_H^* \leftarrow OptimalHumanAction(\Phi_D)$, as per (9.21) .
          $\hat{\beta} = \frac{k}{2\lambda(\|u_H\|^2 - \|u_H^*\|^2)}$ .
          $\hat{\theta} \leftarrow \hat{\theta} - \alpha\frac{\Gamma(\Phi_D, E=1)}{\Gamma(\Phi_D, E=1) + \Gamma(\Phi_D, E=0)}\big(\Phi_D - \Phi(\mathbf{x}_R)\big)$.
          $\mathbf{x}_R \leftarrow TrajOpt(\hat{\theta})$ .
      end if
  end while
```

[8]Note that to enforce the constraint on $||\theta|| = 1$, we can indeed project the resulting $\hat{\theta}'$ onto the unit ball. In practice, because our learning from corrections algorithm separates the β-inference from the θ-inference, this projection is no longer required, but we found it helpful to still constrain the space of Θ to encourage smoothness in the change of the cost function.

Overall, the full algorithm is given in Algorithm 3. The robot begins tracking a trajectory $\mathbf{x}_R$ given by an initial $\hat{\theta}$. Once a human torque u_H is sensed, the robot deforms its trajectory to compute the induced features Φ_D, computes the optimal human action u_H^* using (9.21), and uses it to estimate $\hat{\beta}$ for that input. It then updates $\hat{\theta}$ using the learned distributions $P(\hat{\beta} \mid E = i), \forall i \in \{0, 1\}$, and updates its tracked trajectory $\mathbf{x}_R$. For more practical details on how replanning works, and how to set various hyperparameters, consult Sec. 9.8.2.

9.4.3 Examples

As in Section 9.3, we now illustrate some examples to help lay out some of the setup we will present in our actual experiments in Sections 9.5 and 9.6. We provide intuition for how the estimators of β and θ work when we have a perfectly specified objective space and a misspecified objective space. In all of the examples, the robot reasons about the previously described distance from the table feature. What changes is the feature for which the human decides to provide corrections.

We look at two situations: the human may correct the relevant feature and push the robot closer to the table, or she might provide an poorly-explained input to keep the coffee mug upright. Fig. 9.6 illustrates the two scenarios and contrasts our estimated-β approach to the state of the art fixed-β approach that uses (9.31).

On the top we present the fixed-β method and its performance with both the well-explained and the poorly-explained input. When the input is well explained, the left image shows that the robot learns from the interactions and converges close to the true $\theta = 1$. However, when the input is poorly explained on the right, the robot incorrectly learns fictitious θ values and produces oscillatory behavior.

In the bottom row of Fig. 9.6 we present our described estimated-β method. In the case of well-explained inputs, the value for $\hat{\beta}$ increases, allowing θ to grow up to the real value $\theta = 1$. The method has the same behavior as the state of the art. However, more importantly, in the case of poorly-explained input, our method immediately estimates low $\hat{\beta}$ values, which allows it to significantly reduce unintended learning as compared to the state of the art.

This figure illustrates how situational confidence estimation can aid the robot when the human input is poorly explained. We stress that although our method does not allow the robot to magically learn the correct behavior that the user desires, it greatly reduces unintended learning and undesired behaviors.

9.5 Case Studies

Equipped with our algorithmic approaches to situational confidence estimation, we now consider two case studies in learning from demonstrations and corrections using real human input on a 7-DoF robot manipulator.

9.5.1 Demonstrations

We collected human demonstrations of household motion planning tasks and performed our situational confidence inference offline. We recruited 12 people to physically interact with a JACO 7-DoF robotic arm and analyzed 4 common cases that can arise in the context of personal robotics learning.

For all the experiments in this section, we asked the participants to provide demonstrations with respect to a feature of interest, which the robot might (well-explained) or might not (poorly-explained) have in its hypothesis space. Some of the features that the humans had to prioritize include: distance of the end effector from the table, distance from the person, or distance from the end-effector to a laptop placed on the table.

Before giving any demonstrations, each person was allowed a period of training with the robot in gravity compensation mode to get accustomed to interacting with the robot. When collecting human demonstrations, participants were asked to move the robot arm holding a cup of coffee from the upper shelf of a cupboard to right above the table, across a laptop.

After collecting all demonstrations, we designed the robot's hypothesis space for inference purposes. In all four scenarios that we will illustrate, the robot reasons over the same three features as in (9.14): E, T, and L. Although the robot always knows about these features, the demonstrations may have been given relative to different (and potentially unmodeled) features.

Throughout our scenarios, we tested two hypotheses:

H1. *If the human input is well-explained, our inference procedure places high probability on the correct θ hypothesis, with a high situational confidence β.*

H2. *If the human input is poorly-explained, our inference procedure does not place high probability on any θ hypothesis and is uniform over all hypotheses with low situational confidence β.*

To test these hypotheses, we looked at the resulting inferred belief. Given the demonstrations and a parametrization of the cost function, we first updated the belief over the weight and situational confidence parameters for each single demonstration, $b_{single}(\theta, \beta)$. This gives insights into how a single demonstration can affect the robot's inference procedure.

Next, we used all 12 human demonstrations to obtain a probability distribution over the weight and confidence measures, $b_{all}(\theta, \beta)$ for each scenario. By using multiple demonstrations as evidence about the cost and the situational confidence parameter, we see how in some scenarios multiple demonstrations can help improve confidence in the θ estimation.

We now present experimental results in two scenarios that support our above hypotheses.

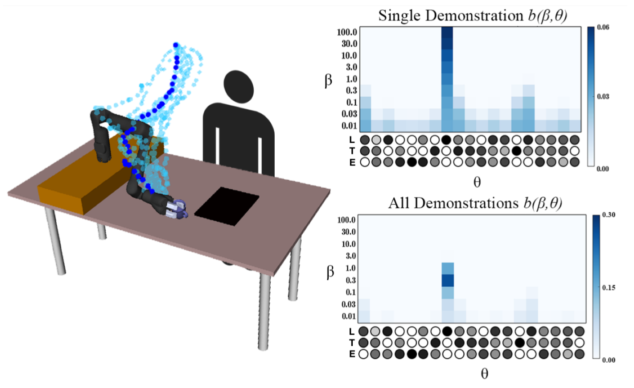


Figure 9.7: (Left) Human demonstrations avoiding the laptop. (Right) Upper distribution is the posterior belief for the highlighted blue demonstration. Since the robot has the laptop feature in its hypothesis space, this demonstration induces a high β on the correct $\theta = [0, 0, 1]$. Below, when considering all the demonstrations, the inference procedure converges to a slightly lower β value due to the suboptimality of some of the demonstrations in the dataset.

Well-specified objective space

Here we consider a scenario where the robot and the human share the same hypothesis space, i.e. the robot's model is well specified. The participants were instructed to avoid spilling the coffee over the laptop by providing a demonstration where the robot's end-effector is away from the electronic device. Here, the feature of interest was the distance from the laptop which was in the robot's hypothesis space: the demonstration would be well explained as long as the demonstration maintained a distance of at least L meters away from the center of the laptop.

On the left of Fig. 9.7 we visualize all 12 recorded demonstrations and the experimental setup. Note that most participants had an easy time providing demonstrations which avoided the laptop. Indeed, we noticed that 10 out of the 12 demonstrations resulted in high situational confidence and a probability distribution similar to the one at the top right of Fig. 9.7. Here, the θ vector that has largest weight on the third feature (distance from the laptop) is correctly inferred to have high β value. This signals that the robot is highly confident the person provided a demonstration that avoids the laptop, which is correct and supports our hypothesis H1.

Another interesting observation is that the situational confidence over all 12 demonstrations together is lower than in the case of the single optimal demonstration highlighted in blue (peak at around 1.0 instead of 100.0)[9]. This is due to the two noisy demonstrations that came too close to the laptop. When working with non-expert users, it is inevitable that such imperfect demonstrations will arise. However, despite the challenge of noisy

[9]In the lower right belief in Fig. 9.7, note from the colorbar values that the probability mass is more peaked than in the case of a single demonstration. This confirms our intuition that the robot's certainty in the hypothesis is enhanced the more demonstrations supporting that hypothesis it receives.

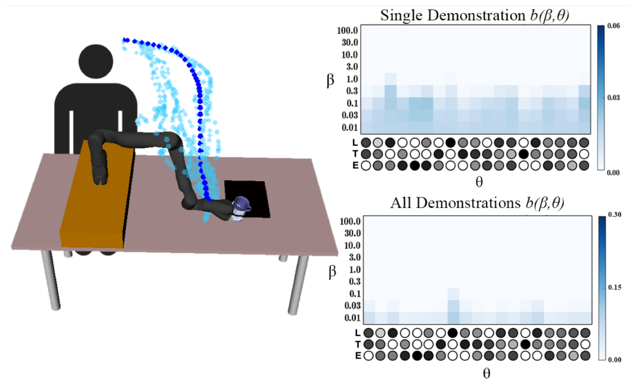


Figure 9.8: (Left) Human demonstrations avoiding the user's body. (Right) Upper distribution is the posterior belief $b(\beta, \theta)$ for the highlighted demonstration. Since the robot's model does not include distance to the user's body, none of the robot's hypotheses can explain the demonstration, as reflected in the higher probabilities on low βs for all θs. After performing inference on all the demonstrations, the distribution in the lower right plot shows even more probability mass on the lowest situational confidence values.

and/or erroneous demonstrations, the algorithm recovers the correct θ hypothesis with a relatively high β, supporting H1 once again.

Misspecified hypothesis space

We look at the opposite scenario: the robot and the human do not share the same hypothesis space and the robot's model is clearly misspecified.

Participants were instructed to move the robot from start to end while also keeping the robot's hand away from their body to avoid spilling coffee on their clothes. Since the robot's cost function does not include any notion of distance to humans, the demonstrations should appear poorly explained relative to the robot's model of how humans choose demonstrations.

Fig. 9.8 visualizes all 12 demonstrations as well as the posterior probability distributions for a single highlighted trajectory and for all 12. For both a single demonstration and all of them, in the case of misspecification none of the hypotheses are correct. Thus, the robot infers equally low probability for all θs, with low situational confidence, supporting our hypothesis H2. This signals that the robot is unsure what the person's demonstration referred to, as we expected.

These two examples illustrate cases where our method supports the two hypotheses above. However, there are important limitations that we discuss in the following two scenarios.

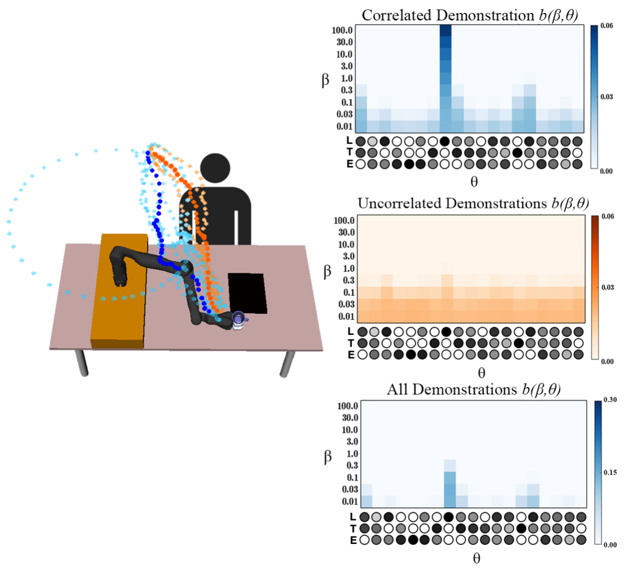


Figure 9.9: (Left) Human demonstrations avoiding the user's body. The blue cluster is correlated with the feature describing distance from the laptop. The orange cluster is uncorrelated. (Right) The top distribution is the posterior belief $b(\beta,\theta)$ for the highlighted blue correlated demonstration. Notice that the hypothesis that puts all weight on avoiding the laptop $\theta = [0, 0, 1]$ dominates the distribution. Meanwhile, the posterior belief for the highlighted orange demonstration indicates low situational confidence in all hypotheses. The bottom distribution shows that when combining all demonstrations, the robot continues to have low situational confidence although the laptop hypothesis has slightly higher β.

Feature correlation

The past two examples demonstrate clear instances when the robot's objective space is either well specified or misspecified. However, often times situations will be more ambiguous. For example, although the human input may refer to a feature that is nonexistent in the robot's hypothesis space, the robot may know about a feature that is correlated to the one the human is trying to affect. In this next scenario, we investigate how such feature correlation influences the situational confidence estimates.

We asked the participants to move the robot from the same start and end as before, while keeping the cup in the robot's end-effector away from their body to avoid spilling coffee on their clothes. The setup is similar to the poorly-explained demonstration in the previous scenario, only that now the human starts in a different initial position.

Visualizations of the 12 demonstrations in Fig. 9.9 showcase that although all demonstrations move the cup away from the person, some of them (depicted in blue) also maintain a good distance away from the laptop. Hence, even though the human was trying to teach the robot to stay away from their body, the robot interprets the human's demonstrations as a signal to stay away from the laptop. Thus, we say that the distance from human and distance from laptop features are *correlated*.

When looking at the top-right posterior probability in Fig. 9.9, the distribution over

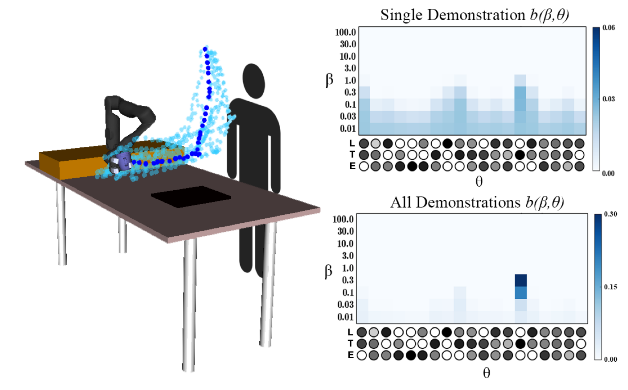


Figure 9.10: (Left) Human demonstrations keeping the cup in the end-effector close to the table. (Right) Because it is difficult for the person to give a good demonstration, the top posterior does not have a clearly defined peak for one particular hypothesis, although several θs are favored. In the bottom distribution, we notice that when presented with all 12 demonstrations, the robot can more clearly infer the correct hypothesis for the distance to the table, $\theta = [0, 1, 0]$.

θ, β shows that our algorithm infers a high situational confidence for the θ that fully considers the distance from the laptop. Thus, even if the human input does not pertain to a feature in the robot's hypothesis space, in some cases the demonstration can still be explained via correlated features in the robot's hypothesis space. This observation does not support H2 and is clearly a limitation of our method.

However, the orange cluster of demonstrations in Fig. 9.9, showcase the fine line between demonstrations that induce a feature correlation and those that do not. The orange demonstrations clearly ignore the laptop and simply take the shortest path to the end goal while avoiding the human's body. As we can see in the orange probability distribution, our method infers a uniform distribution over all θ hypotheses, with a focus on the lowest situational confidence values, backing H2.

These two clusters highlight that our method infers reasonable θ, β values even in the case of feature correlation. The robot either infers a good θ to perform its original task through the means of another feature, or it has low confidence in understanding the input.

When we look at the posterior distribution that results from all 12 demonstrations, the bottom-right part of the figure shows that, due to the correlation in the blue cluster, there is increased probability on the θ that considers fully the distance from the laptop. However, due to the ambiguity of the orange cluster, the situational confidence is not as high as it would be in a well-explained case (see Fig. 9.7).

Feature engineering

Many of the cost function features we considered so far have been intuitive to provide demonstrations for. However, some cost functions may be particularly challenging or unintuitive for human users. Two extreme examples of this could be features learned using complex function approximators or unintuitive features like minimizing the total energy of a system.

In our scenario, the feature users have a difficult time providing good demonstrations for is the distance between the robot's hand and the table along the trajectory. Since the feature was designed as the sum of distances to table for all waypoints in the trajectory, the optimal demonstration immediately moves the end-effector to the table and then keeps it right above the tabletop for the rest of the path, as seen in Fig. 9.3(a). This limitation does not support H1.

However, this mathematical optimum does not necessarily align with how human users interpret the best behavior for this task. In our experiments, most users gradually bring the robot's hand closer to the table, rather than pushing it down immediately, for a more smooth and natural motion (see left in Fig. 9.10). These demonstrations thus appear noisy and sub-optimal with respect to the robot's model and make it difficult to infer the true θ from a single demonstration.

This phenomenon is reflected more clearly when we look at the top-right belief distribution in Fig. 9.10. Although the distribution for the highlighted blue demonstration has some peaks around hypotheses that strongly favor the feature responsible for distance to the table, it is not nearly as clearly defined as it should be for a well-explained demonstration (see Fig. 9.7).

However, when the robot gathers evidence from multiple demonstrations, the algorithm does manage to figure out that this is the feature that people were optimizing for. The bottom right plot in Fig. 9.10 illustrates that, once again, having more input samples eventually leads our algorithm to converge to a strong probability for the right θ with a reasonably high β. Although our method cannot back H1 when inferring the objective from a single demonstration, more data leads our algorithm to correctly support H1.

Summary: The four situations presented above illustrate that our two original hypotheses H1 and H2 are supported most of the time (9.5.1, 9.5.1), with some exceptions (9.5.1, 9.5.1). We saw that when the person has a difficult time giving a good demonstration (Section 9.5.1), our method cannot support H1 unless provided with multiple demonstrations, to disambiguate the inherent noise in the user's suboptimal input. Additionally, when the person provides what should be a poorly-explained demonstration (Section 9.5.1), feature correlation might lead the inference to falsely detect θs corresponding to that input, contradicting H2. However, we observed that when given more demonstrations, our algorithm can attribute low situational confidence β if the uncorrelated input is sufficient. More work is needed in this area.

9.5.2 Corrections

We now turn our attention to case where human input is sparse and in the form of intermediate corrections during the robot's task execution. Here we present an offline case study where we analyze how our estimates of $\hat{\beta}$ enable us to distinguish if the input is well explained or not to the robot's model of the human. For a full exploration of the real-time updates from human corrections, we conduct an online user study which we later describe in Section 9.6.

We recruited 12 additional individuals to physically interact with the same robotic manipulator. Each participant was asked to intentionally correct a feature (that the robot may or may not have in its hypothesis space): adjusting the distance of the end effector from the table, adjusting the distance from the person, or adjusting the cup's orientation.

During this case study the robot did not attempt to update the feature weights θ and simply tracked a predefined trajectory with an impedance controller [**impedance**]. The participants were instructed to intervene only once during the robot's task execution, in order to record a single physical correction. The resulting trajectories and physical interaction u_H were saved for offline analysis. This setup enabled us to easily analyze the situational confidence of the robot as we changed the robot's hypothesis space.

Next, we ran our approximate inference algorithm using the recorded human interaction torques and robot joint angle information. We measured what $\hat{\beta}$ would have been for each interaction if the robot knew about a given subset of the features. By changing the subset of features for the robot, we changed whether any given human interaction was well *explained* to the robot's hypothesis space.

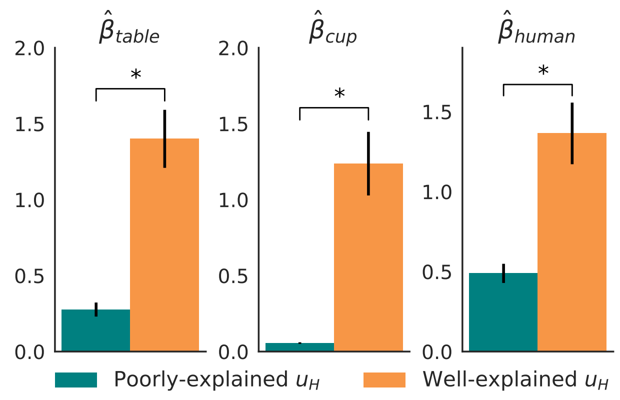
We tested two hypotheses:

> **H1.** *Well-explained interactions result in high $\hat{\beta}$, whereas interactions that change a feature the robot* ***does not*** *know about result in low $\hat{\beta}$ for all features the robot* ***does*** *know about.*

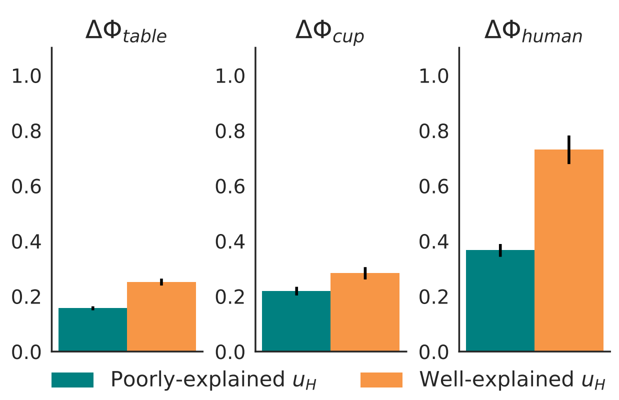
> **H2.** *Not reasoning about well-explained interactions and, instead, indiscriminately learning from every update leads to significant unintended learning.*

We ran a repeated-measures ANOVA to test the effect of whether and input is well explained on our $\hat{\beta}$. We found a significant effect ($F(1, 521) = 9.9093$, $p = 0.0017$): when the person was providing a well-explained correction, $\hat{\beta}$ was significantly higher. This supports our hypothesis H1.

Fig. 9.11(a) plots $\hat{\beta}$ under the well-explained (orange) and poorly-explained (blue) conditions. Whereas the poorly-explained interactions end up with $\hat{\beta}$s close to 0, well-explained corrections have higher mean and take on a wider range of values, reflecting varying degrees of human performance in correcting something the robot knows about. We fit per-feature chi-squared distributions for $P(\hat{\beta} \mid E)$ for each value of E which we will use to infer E and, thus, θ online. In addition, Fig. 9.11(b) illustrates that even for poorly-explained human actions u_H, the resulting feature difference $\Delta\Phi = \Phi(\mathbf{x}_D) - \Phi(\mathbf{x})$ is non-negligible. This supports our second hypothesis, H2, that not reasoning about how well-explained an action is is detrimental to learning performance when the robot receives misspecified updates.



(a) Average β for well-explained and poorly-explained interactions.



(b) Average $\Delta\Phi$ for well-explained and poorly-explained interactions.

Figure 9.11: β values are significantly larger for well-explained actions than for poorly-explained ones. Feature updates are non-negligible even during poorly-explained actions, which leads to significant unintended learning for fixed-β methods.

9.6 User Study on Learning from Corrections

Our case study on corrections suggested that $\hat{\beta}$ can be used as a measure of whether physical interactions are well explained and should be learned from. Next, we conducted an IRB-approved user study to investigate the implications of using these estimates during learning. During each experimental task, the robot began with a number of incorrect weights and participants were asked to physically correct the robot. Locations of the objects and human were kept consistent in our experiments across tasks and users to control for confounds[10]. The planning and inference were done for robot trajectories in 7-dimensional configuration space, accounting for all relevant constraints including joint limits and self-collisions, as well as collisions between obstacles in the workspace and any part of the robot's body.[11]

9.6.1 Experiment design

Independent variables

We used a 2 by 2 factoral design. We manipulated the corrections learning strategy with two levels (fixed-β and estimated-β learning), and also whether the human corrected for features inside (well explained) or outside (poorly explained) the robot's hypothesis

[10]We assume full observability of where the objects and the human are, as the focus of this paper is not sensing.

[11]For video footage of the experiment, see: https://youtu.be/stnFye8HdcU

space. In the fixed learning strategy, the robot updated its feature weights from a given interaction via (9.31) with a fixed β value. In the estimated-β learning strategy, the robot updates its feature weights via (9.30). The offline experiments above provided us an estimate for $P(E \mid \hat{\beta})$ that we used in the gradient update.

Dependent measures - objective

To analyze the objective performance of the two learning strategies, we focused on comparing two main measurements: the length of the $\hat{\theta}$ path through weight space as a measurement of the learning process, and the regret in feature space measured by $|\Phi(\mathbf{x}_{\theta^*}) - \Phi(\mathbf{x}_{actual})|$. Longer $\hat{\theta}$ paths should indicate a learning process that oscillates, whereas shorter paths suggest smoother learning curves. On the other hand, high regret implies that the learning method did not converge to a good objective θ, whereas low regret indicates better learning.

Dependent measures - subjective

For each condition, we administered a 7-point Likert scale survey about the participant's interaction experience (Table 9.1). We separate the survey into 3 scales: task completion, task understanding, and unintended learning.

Hypotheses

We tested four hypotheses:

H1. *On tasks where humans try to correct* ***inside*** *the robot's hypothesis space (well-explained corrections), inferring situational confidence is not inferior to always assuming high situational confidence.*

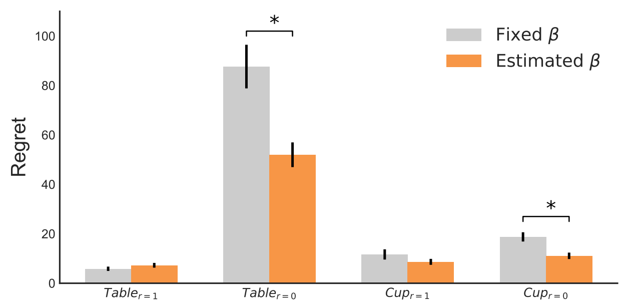
H2. *On tasks where humans try to correct* ***outside*** *the robot's hypothesis space (poorly-explained corrections), inferring situational confidence reduces unintended learning.*

H3. *On tasks where they tried to correct* ***inside*** *the robot's hypothesis space, participants felt like the two methods performed the same.*

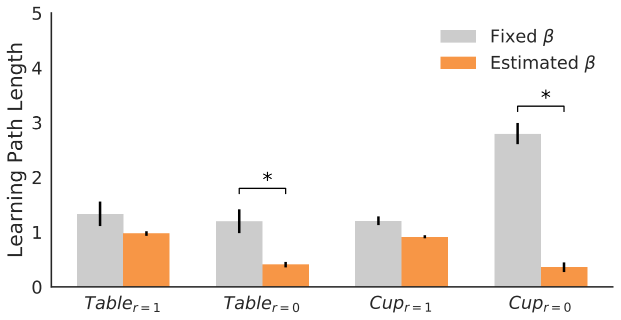
H4. *On tasks where they tried to correct* ***outside*** *the robot's hypothesis space, participants felt like our estimated-β method reduced unintended learning.*

Tasks

We designed 4 experimental household motion planning tasks for the robot to perform in a shared workspace. Similarly to the case studies, for each experimental task, the robot carried a cup from a start to end pose with an initially incorrect objective. Participants were instructed to physically intervene to correct the robot's behavior during the task.



(a) Regret averaged across subjects.



(b) $\hat{\theta}$ learning path length averaged across subjects.

Figure 9.12: Comparison of regret and length of $\hat{\theta}$ learning path through weight space over time (lower is better).

In Tasks 1 and 2, the robot's default trajectory took a cup from the participant and put it down on the table, but carried the cup too high above the table. In Tasks 3 and 4, the robot also took a cup from the human and placed it on the table, but this time it initially grasped the cup at the wrong angle, requiring human assistance to correct end-effector orientation to an upright position. For Tasks 1 and 3, the robot knew about the feature the human was asked to correct for ($E = 1$) and participants were told that the robot should be compliant. For Tasks 2 and 4, the correction was poorly explained ($E = 0$) and participants were instructed to correct any additional unwanted changes in the trajectory.

Participants

We used a within-subjects design and randomized the order of the learning methods during experiments. We recruited 12 participants (6 females, 6 males, aged 18-30) from the campus community, 10 of which had technical background. None of the participants had experience interacting with the robot used in our experiments.

Procedure

Every participant was assigned a random ordering of the two methods, and performed each task without knowing how the underlying methods work. One challenge in performing and evaluating our experiment was that different participants may have different internal preferences for how a task should be performed. In order to have a consistent notion of ground-truth preferences, we fixed the true objective (e.g. how far the cup should be from the table) for each task. At the beginning of each task, the participant was first shown the incorrect default trajectory that they must correct, followed by the ground-truth desired trajectory they should teach the robot. This allows us to focus only on how well each algorithm infers objectives from human input, versus trying to addi-

	Questions	Cronbach's α	F-Ratio	p-value
task	The robot accomplished the task in the way I wanted. The robot was NOT able to complete the task correctly.	0.94	0.88	0.348
understand	I felt the robot understood how I wanted the task done. I felt the robot did NOT know how I wanted the task done.	0.95	0.55	0.46
unintend	I had to undo corrections that I gave the robot. The robot wrongly updated its understanding about aspects of the task I did not want to change. After I adjusted the robot, it continued to do the other parts of the task correctly. After I adjusted the robot, it incorrectly updated parts of the task that were already correct.	0.91	9.15	**0.0046**

Table 9.1: Results of ANOVA on subjective metrics collected from a 7-point Likert-scale survey.

tionally estimate the unique ground-truth human objective of each participant. Then the participant performed a familiarization round, followed by two recorded experimental rounds. After answering the survey, the participant repeated the procedure for the other method.

9.6.2 Analysis

Objective

We ran a repeated-measures factorial ANOVA with learning strategy and input quality (well or poorly explained) as factors for the regret. We found a significant main effect for the method ($F(1, 187) = 7.8, p = 0.0058$), and a significant interaction effect ($F(1, 187) = 6.77, p = 0.0101$). We ran a post-hoc analysis with Tukey HSD corrections for multiple comparisons to analyze this effect, and found that it supported our hypotheses. On tasks where corrections were poorly explained, the estimated-β method had significantly lower regret ($p = 0.001$); on tasks where corrections were well explained, there was no significant difference ($p = 0.9991$). Fig. 9.12(a) plots the regret per task, and indeed the estimated-β method was not inferior on tasks 1 and 3, and significantly better on tasks 2 and 4.

For the length of the $\hat{\theta}$ path through weight space metric, the factorial ANOVA analysis found a significant main effect for the method ($F(1, 187) = 76.43, p < 0.0001$), and a significant interaction effect ($F(1, 187) = 33.3, p < 0.0001$). A similar post-hoc analysis with Tukey HSD correction for multiple comparisons also supports our hypotheses. On tasks where corrections were poorly explained, our method had significantly lower average weight paths over time ($p = 0.0025$); on tasks where correction were well explained, however, there was no significant difference ($p = 0.1584$). The same results are supported by Fig. 9.12(b), which plots the average length of $\hat{\theta}$ through weight space per task, and indeed our method was not significantly inferior for tasks 1 and 3, and significantly better on tasks 2 and 4.

Subjective

We ran a repeated measures ANOVA on the results of our participant survey. We find that our method is not significantly different from the baseline in terms of task completion ($F(1,7) = 0.88, p = 0.348$) and task understanding ($F(1,7) = 0.55, p = 0.46$), which supports H3. Participants also significantly preferred the estimated-β method in terms of reducing unintended learning ($F(1,7) = 9.15, p = 0.0046$), which supports H4.

9.7 Discussion

Human guidance is becoming increasingly important as autonomous systems enter the real world. One common way for robots to interpret human input is treating it as evidence about hypotheses in the robot's objective space. Since accounting for all possible hypotheses and situations ahead of time is challenging if not infeasible, in this paper we claim that robots should explicitly reason about how well their given hypothesis space can explain the human inputs.

We introduced the notion of *situational confidence* β as a natural way to measure how much the robot should trust its inputs and learn from them. We presented a general framework for estimating β in conjunction with any task objectives for scenarios where the human and the robot are operating the same dynamical system. We instantiated it for learning from human demonstrations, as well as for learning from corrections, by deriving a close to real-time approximate algorithm. In both settings, we exemplified – via human experiments with a 7-DoF robotic manipulator and a user study – that reasoning about situational confidence does, in fact, assist the robot in better understanding when it cannot explain human input.

There are several important limitations in our work. Perhaps the biggest limitation of all, which we alluded to in the introduction, is that the hypothesis space can be misspecified but the robot can nonetheless explain the input relatively well, thus confusing misspecification for slight noise. This is especially true in more expressive hypothesis spaces, where there might always be some hypothesis that explains the input. This is unfortunately a fundamental problem with detecting misspecification in expressive hypothesis spaces: a single demonstration or a single data point will not be enough. Much like learning cost functions when using such spaces requires much more and diverse data than when using a less expressive space, with detecting misspecification too it will be the case that the robot will require a rich and diverse set of data points. The more data the robot has access to, and the more diversely it is distributed, the less of a chance there is that one wrong hypothesis can explain all the data.

Furthermore, our approach cannot disambiguate between misspecification of the hypothesis space and misspecification of the human observation model, i.e. the Boltzmann model.

Algorithmically, while for corrections we derived a way to handle continuous hypothesis spaces that scales linearly with the dimensionality of the space, for demonstrations we relied on simply discretizing the space. This was sufficient for showcasing the benefit of estimating situational confidence, since for demonstrations this is done offline. However, to scale the method to complex spaces, we need to combine it with state-of-the-art (Bayesian) IRL approaches that rely on Metropolis Hastings sampling, or simply estimate the MLE.

Lastly, our experiments for both demonstrations and corrections are limited to a simple motion planning task with a cost function that depends on only a few features. We do not show how the method would degrade, both under ideal as well as under approximate inference.

In subsequent work, we hope to address some of these limitations. We are also interested in an extension to sequential time-dependent inputs, where the person could change their mind about what objective is important to them. Additionally, we want to explore ways of handling misspecification other than reducing learning, such as switching to a more expressive hypothesis space (but demanding more data and computation) whenever the situational confidence is very low for all θs. Finally, we are excited to showcase our work on other coupled dynamical systems, such as autonomous vehicles.

9.8 Practical Considerations

9.8.1 Demonstrations

Discretizing Θ and $\mathcal{B}$ in (9.13)

For the Θ discretization, we chose vectors in the unit sphere, as discussed in Section 9.2.2. For practical purposes, we restricted the θ components to be positive due to our task features and the capabilities of our trajectory optimizer; in general, learning from demonstrations should be restricted to norm 1, not necessarily to the positive quadrant. In both our examples in Section 9.3 and experiments in Sections 9.5, each θ_i component was allowed to take values 0, 0.5, or 1. Since we used 3 features, θ's dimensionality was 3, leading to a possible set Θ equivalent to the 3-fold Cartesian product of the values above. After normalizing to norm 1, we were left with 19 unique θ vectors in Θ, weighing the three features in different proportions, as shown in Figures 9.3, 9.7, 9.8, 9.9, and 9.10. Our discretization scheme ensured an approximately uniform sampling on the positive quadrant of the unit sphere.

To discretize situational confidence, we found it sufficient to cover

$$\beta \in \{0.01, 0.03, 0.1, 0.3, 1.0, 3.0, 10.0, 30.0, 100.0\},$$

the log-scale space, similarly to [77, 74]. For different tasks, a similar discretization should suffice because what matters is β's relative magnitude for identifying misspecification, not its absolute one. We suggest calibrating the threshold ϵ in (9.6) using a few simulated trajectories like the ones in Fig. 9.3.

9.8.2 Corrections

Planning and Replanning

We use TrajOpt [**trajopt**] to plan and replan robot trajectories. We set up the trajectory optimization problem to plan a path that minimizes a cost function of the form of (9.15). Given different features Φ and weights θ on these features, different optimal paths may be found. Additionally, we constrain the optimization to plan a path between a pre-specified start and goal locations, while avoiding collisions with the objects in the environment (table, laptop, or human). The total time of the trajectory is fixed, but the actual length can differ. That means that the robot moves faster for longer trajectories, and slower for shorter ones.

When the experiment starts, the robot plans an initial path from start to goal, using the initial weights θ. When a human push happens, the robot measures the instantaneous deviation, which deforms the trajectory via the impedance controller. Without learning, the robot would resume tracking its original trajectory. However, we use the human input to update θ according to (9.30), which the robot's planner uses to compute a new trajectory that the robot can follow instead. In a perfect world, this entire process would happen at 60Hz. In practice, however, the trajectory optimizer's computation lasts longer. As such, once a push is registered, the robot starts listening for following torque signals only after the update is complete.

Imagine this process in the context of a typical user experience. Once the person begins pushing, the robot instantly starts updating θ and optimizing the new induced path. While the person is applying their correction, the planner eventually finishes its computation and passes the updated trajectory to the robot controller. The user can immediately feel that the robot changed course and stops intervening.

Solving (9.21)

We used SLSQP, an off-the-shelf sequential quadratic programming package [125], to solve (9.21). In practice, the method can fail to return a good result if the initialization is bad. We found that if we initialize the minimization with a guess that does not satisfy the constraint (e.g. 0), it returns a reasonable estimate of the true u_H^*.

Sensitivity Analysis

Both (9.24) and (9.30) rely heavily on hyperparameters λ and ν. Here, we discuss how to set them.

Setting λ affects the magnitude of the resulting estimated situational confidence $\hat{\beta}$ in (9.24). This magnitude plays an important role when later estimating θ via (9.30) because it affects $P(E \mid \hat{\beta})$. However, note that to compute this probability we use $P(\hat{\beta} \mid E)$, which is an entirely data-driven empirical distribution, where the observed $\hat{\beta}$ is also computed via (9.24). As such, we are not relying on absolute magnitudes of the estimated situational confidence but on relative ones. Therefore, the choice of the hyperparameter λ does not affect our method's estimates as long as they are computed with the same hyperparameter that is used for learning $P(\hat{\beta} \mid E)$.

In the case of precision ν in (9.25), how spread out the Gaussian noise centered around $\Phi(\mathbf{x}_R)$ is affects the denominator in (9.30). When $\nu \to 0$, the $\Gamma(\Phi_D, E = 0)$ term in the denominator goes to 0, which means that (9.30) reduces to (9.31): our method always learns and never identifies misspecification. On the other hand, when $\nu \to \infty$, we can use the L'Hospital rule to see that $\Gamma(\Phi_D, E = 0) \to 0$ as well, as long as $||\Phi_D - \Phi(\mathbf{x}_R)||^2 \neq 0$, which is true unless there is no correction to deform $\mathbf{x}_R$, in which case we do not need to update θ at all. Therefore, it is important that ν is set not too high and not too low in order for our method to work properly.

The best practice for setting ν also involves using the offline data calibration from Section 9.5.2. To calibrate properly, after computing the empirical $P(\hat{\beta} \mid E)$ distribution, when $E = 0$ the updated θ should not change much, whereas when $E = 1$ the θ parameter should change appropriately.

Without the offline data calibration in Section 9.5.2, both λ and ν affect the θ and β estimation, and can have profound effects on the efficacy of our method. Unfortunately, we cannot do this calibration automatically yet, which is a limitation of our work, and we leave it for future research.

Trajectory Deformation Parameter Choice

When deforming the robot's trajectory given a human interaction, there are many choices of the deformation matrix A and the deformation magnitude parameter μ. A can be an explicit design choice (for example, constructing A from a finite differencing matrix [19]), can be solved for via an optimization problem which penalizes the undeformed trajectory's energy, the work done by the trajectory deformation to the human, and variation's total jerk as in [146], or can even be learned from human data [104]. The magnitude of the deformation μ can also be tuned for best performance, for example to be robust to the rate at which deformations occur (see [144] for more details).

9.9 Laplace Approximation in Equation (9.19)

Let the cost function in the model in (9.19) be denoted by:

$$C_{\Phi_D}(\bar{u}) = \lambda \|\bar{u}\|^2 + \kappa \|\Phi(\bar{\mathbf{x}}_D) - \Phi_D\|^2, \tag{9.32}$$

for an observed Φ_D.

First, our cost function can be approximated to quadratic order by computing a second order Taylor series approximation about the optimal human action u_H^* (obtained via the constrained optimization in 9.21):

$$\begin{aligned} C_{\Phi_D}(\bar{u}) \approx\ & C_{\Phi_D}(u_H^*) + \nabla C_{\Phi_D}(u_H^*)^\top(\bar{u} - u_H^*) \\ & + \frac{1}{2}(\bar{u} - u_H^*)^\top \nabla^2 C_{\Phi_D}(u_H^*)(\bar{u} - u_H^*) \ . \end{aligned} \tag{9.33}$$

Since $\nabla C_{\Phi_D}(\bar{u})$ has a global minimum at u_H^* then $\nabla C_{\Phi_D}(u_H^*) = 0$ and the denominator of Equation 9.19 can be rewritten as:

$$\int_{\mathcal{U}} e^{-\beta C_{\Phi_D}(\bar{u})} d\bar{u} \approx e^{-\beta C_{\Phi_D}(u_H^*)} \int_{\mathcal{U}} e^{-\frac{1}{2}(\bar{u}-u_H^*)\beta\nabla^2 C_{\Phi_D}(u_H^*)(\bar{u}-u_H^*)} d\bar{u} \ . \tag{9.34}$$

Since $\beta\nabla^2 C_{\Phi_D}(u_H^*) > 0$ for $u_H^* \neq 0$, the integral is in Gaussian form, which admits a closed form solution:

$$\int_{\mathcal{U}} e^{-\beta C_{\Phi_D}(\bar{u}_H)} d\bar{u}_H \approx e^{-\beta C_{\Phi_D}(u_H^*)} \sqrt{\frac{2\pi^k}{\beta^k |H_{u_H^*}|}} \ ,$$

where $H_{u_H^*} = \nabla^2 C_{\Phi_D}(u_H^*)$ denotes the Hessian of C_{Φ_D} at u_H^*. Replacing $C_{\Phi_D}(\bar{u}_H)$ with the expanded cost function, we arrive at the final approximation of the observation model:

$$P(u_H^t \mid x^0, \mathbf{u}_R; \Phi_D, \beta) \approx \frac{e^{-\beta\lambda(\|u_H^t\|^2)}}{e^{-\beta(\lambda\|u_H^*\|^2 + \kappa\|\Phi(\mathbf{x}_D^*) - \Phi_D\|^2)}} \sqrt{\frac{\beta^k |H_{u_H^*}|}{2\pi^k}} \ . \tag{9.35}$$

Chapter 10

Conclusion and Future Work

From autonomous cars in cities to mobile manipulators at home, I aim to design robots that interact with people. While robot interaction with people is necessary, it is also a potential mechanism for safety issues such as miscoordination, collision, end-user discomfort, erroneous robot learning, or long-term unintended consequences. While my thesis primarily focused on traditional collision-avoidance notions of safety, close collaboration with people demands more than just collision-avoidance, raising the question: *what is the right notion of safety?* This was apparent during my work on robot learning from physical human interactions [19, 18, 148]: after consistently misinterpreting my feedback during a household cleaning task, the robot erroneously learned to move my coffee mugs at an angle, resulting in spilled coffee and miscoordination.

Thus, future work needs to rethink our notions of safety to capture more subtle aspects of human-robot interaction. This demands formalizing novel safety concepts related to robot alignment with human values and the ability to optimize for human preferences. Ultimately, safe human-robot interaction will require robots to not only reason about the limitations of their human models, but also their perception capabilities and state-space representations. I am excited to see future work towards robots that interact safely and intelligently despite imperfect human models: assistive robots that use perceptual data to augment their understanding of human feedback, autonomous cars that understand the long-term effect of model errors on their decisions, and robotic manipulators that leverage diverse sensing modalities to gently but intentionally initiate physical interaction with people. In this section, I briefly outline a few future research directions towards such reliable human-robot interaction.

Safety for robots *learning from* and *coordinating with* people

Robot safety around humans has been predominantly focused on collision-avoidance. However, robots that learn from and coordinate with people demand new, more nuanced notions of safety.

Safe robot *learning from* people. My work on robots learning from physical human interaction [19, 18] revealed how robots can erroneously learn from human input if the human teaches the robot about an aspect of a task it *does not* knows about. I extended my confidence-aware human models to the learning from demonstration domain [34], enabling robots to prevent unintended learning from human interactions when they do not understand human input. Building on this foundation of safety analysis, human modelling, and machine learning, I plan to develop novel formalisms for safety in collaborative robot learning from people. These formalisms would, for example, enable robots to assess which data to learn from (e.g., teaching human inputs), which data to ignore (e.g., adversarial human inputs), and how to strategically gather additional information from nearby humans.

Safe robot *coordination with* people. Even when the robot is not explicitly learning from humans, we still need new methods for safety in robot coordination with humans. For example, here safety can mean ensuring that robots do not influence nearby humans negatively and behave too "selfishly" when pursuing their own objectives. I am particularly interested in approaching these problems through the lens of game-theory, which rigorously couples the influence between the human, robot, and their respective objectives.

Confidence-awareness for high-capacity learned human models

With more available human data, robots increasingly depend on large learned human models; in autonomous driving, function approximators like neural networks are widely used human motion predictors. These high-capacity human models bring not only significant opportunities, but also challenges.

Active data gathering. Much of the available human data exhibits average-case human behavior (e.g., straight highway driving). This limits data-driven models because they cannot extrapolate to the breadth of interactions with humans. Here, I aim to develop methods for actively and efficiently querying humans for supervisory input; this input can augment or label datasets used for robot learning.

Safe robot decision-making. I plan to develop algorithms for quantifying—and improving—the limitations of robot decision-making when relying on complex, data-driven human models. Towards this vision, I'm excited about developing methods for bringing confidence-awareness to high-capacity human models by, for example, designing algorithms which detect out-of-distribution human behavior.

A "full-stack" approach to safe human-robot interaction

Safe human-robot interaction should ultimately account for the robot's perception capabilities, state-space representations, and dynamics. My preliminary work on adaptive model selection is a step towards robots explicitly reasoning about how model fidelity affects planning, proving useful for human motion prediction and robot dynamics estimation [176]. Beyond this, I am excited about developing robot algorithms which can learn to interact with humans in a perception-aware way. For example, instead of hand-engineered state-space models, dynamics models, and feature spaces, the robot could automatically extract task and state representations from camera data that are relevant for human interaction. In fact, with the advent of deep learning, this approach is becoming increasingly popular (for example, but not limited to, [225, 70]). These perceptually-aware representations can also serve as a starting point for *aligning* the robot and human's understanding of the task [32]. However, this also raises the question about the *quality* of these automatically extracted representations. For instance, imagine that a robot's representation implicitly understands how the location of your hand matters for handovers, but misses how your body orientation also matters for predicting your behavior. However, understanding *the limitations* of what the robot understands is now an increasing concern. This underscores the need for new safety analysis of these perceptually-aware representations themselves. These are just small examples of how developing robot algorithms that leverage real sensors can open up new research directions for safe human-robot interaction.

Bibliography

[1] Nematollah Ab Azar, Aref Shahmansoorian, and Mohsen Davoudi. "From inverse optimal control to inverse reinforcement learning: A historical review". In: *Annual Reviews in Control* 50 (2020), pp. 119–138.

[2] Pieter Abbeel, Adam Coates, and Andrew Y Ng. "Autonomous helicopter aerobatics through apprenticeship learning". In: *The International Journal of Robotics Research* 29.13 (2010), pp. 1608–1639.

[3] Pieter Abbeel and Andrew Y Ng. "Apprenticeship learning via inverse reinforcement learning". In: *Proceedings of the twenty-first international conference on Machine learning*. 2004, p. 1.

[4] Mohamadreza Ahmadi et al. "Control theory meets POMDPs: A hybrid systems approach". In: *arXiv preprint arXiv:1905.08095* (2019).

[5] Ali Ahmadzadeh et al. "Multi-vehicle path planning in dynamically changing environments". In: *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*. IEEE. 2009, pp. 2449–2454.

[6] Baris Akgun et al. "Keyframe-based learning from demonstration". In: *International Journal of Social Robotics* 4.4 (2012), pp. 343–355.

[7] Matthias Althoff and John M Dolan. "Set-based computation of vehicle behaviors for the online verification of autonomous vehicles". In: *2011 14th International IEEE Conference on Intelligent Transportation Systems (ITSC)*. IEEE. 2011, pp. 1162–1167.

[8] Heni Ben Amor et al. "Interaction primitives for human-robot cooperation tasks". In: *Robotics and Automation (ICRA), 2014 IEEE International Conference on*. IEEE. 2014, pp. 2831–2837.

[9] Galen Andrew and Jianfeng Gao. "Scalable training of l 1-regularized log-linear models". In: *Proceedings of the 24th international conference on Machine learning*. 2007, pp. 33–40.

[10] Georges S. Aoude et al. "Probabilistically Safe Motion Planning to Avoid Dynamic Obstacles with Uncertain Motion Patterns". In: *Auton. Robots* 35.1 (2013), pp. 51–76. DOI: 10.1007/s10514-013-9334-3.

[11] Brenna D Argall et al. "A survey of robot learning from demonstration". In: *Robotics and Autonomous Systems* 57.5 (2009), pp. 469–483.

[12] Brenna D. Argall, Eric L. Sauser, and Aude G. Billard. "Tactile Guidance for Policy Adaptation". In: *Found. Trends Robot* 1.2 (Feb. 2011), pp. 79–133. ISSN: 1935-8253. DOI: 10.1561/2300000012.

[13] Haoyu Bai et al. "Intention-aware online POMDP planning for autonomous driving in a crowd". In: *Robotics and Automation (ICRA), 2015 IEEE International Conference on*. IEEE. 2015, pp. 454–460.

[14] Andrea Bajcsy et al. "A Robust Control Framework for Human Motion Prediction". In: *IEEE Robotics and Automation Letters* 6.1 (2020), pp. 24–31.

[15] Andrea Bajcsy et al. "A Scalable Framework For Real-Time Multi-Robot, Multi-Human Collision Avoidance". In: *arXiv preprint arXiv:1811.05929* (2018).

[16] Andrea Bajcsy et al. "An efficient reachability-based framework for provably safe autonomous navigation in unknown environments". In: *2019 IEEE 58th Conference on Decision and Control (CDC)*. IEEE. 2019, pp. 1758–1765.

[17] Andrea Bajcsy et al. "Analyzing human models that adapt online". In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2021, pp. 2754–2760.

[18] Andrea Bajcsy et al. "Learning from Physical Human Corrections, one Feature at a Time". In: *ACM/IEEE International Conference on Human-Robot Interaction (HRI)*. 2018, pp. 141–149.

[19] Andrea Bajcsy et al. "Learning robot objectives from physical human interaction". In: *Conference on Robot Learning (CoRL)*. 2017.

[20] Chris L Baker and Joshua B Tenenbaum. "Modeling human plan recognition using Bayesian theory of mind". In: *Plan, activity, and intent recognition: Theory and practice* 7 (2014), pp. 177–204.

[21] Chris L Baker, Joshua B Tenenbaum, and Rebecca R Saxe. "Goal inference as inverse planning". In: *Cognitive Science Society*. 2007.

[22] Frank J Balbach and Thomas Zeugmann. "Recent developments in algorithmic teaching". In: *International Conference on Language and Automata Theory and Applications*. 2009, pp. 1–18.

[23] Tirthankar Bandyopadhyay et al. "Intention-aware motion planning". In: *Algorithmic Foundations of Robotics X*. Springer, 2013, pp. 475–491.

[24] Somil Bansal and Claire J Tomlin. "Deepreach: A deep learning approach to high-dimensional reachability". In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2021, pp. 1817–1824.

[25] Somil Bansal et al. "Hamilton-Jacobi reachability: A brief overview and recent advances". In: *Decision and Control (CDC), 2017 IEEE 56th Annual Conference on*. IEEE. 2017, pp. 2242–2253.

[26] Somil Bansal et al. "Safe sequential path planning of multi-vehicle systems under presence of disturbances and imperfect information". In: *Proc. Amer. Control Conf.* 2017, pp. 1–8.

[27] Somil Bansal* et al. "A Hamilton-Jacobi reachability-based framework for predicting and analyzing human motion for safe planning". In: *ICRA*. 2020.

[28] Tamer Başar and Geert Jan Olsder. *Dynamic noncooperative game theory*. SIAM, 1998.

[29] Fethi Belkhouche. "Reactive path planning in a dynamic environment". In: *IEEE Transactions on Robotics* 25.4 (2009), pp. 902–911.

[30] Richard Bellman. "The theory of dynamic programming". In: *Bulletin of the American Mathematical Society* 60.6 (1954), pp. 503–515.

[31] Dimitri Bertsekas. *Dynamic programming and optimal control: Volume I*. Vol. 1. Athena scientific, 2012.

[32] Andreea Bobu et al. "Learning perceptual concepts by bootstrapping from human queries". In: *arXiv preprint arXiv:2111.05251* (2021).

[33] Andreea Bobu et al. "Learning under Misspecified Objective Spaces". In: *2nd Annual Conference on Robot Learning, CoRL 2018, Zürich, Switzerland, 29-31 October 2018, Proceedings*. 2018, pp. 796–805. URL: http://proceedings.mlr.press/v87/bobu18a.html.

[34] Andreea Bobu et al. "Quantifying hypothesis space misspecification in learning from human–robot demonstrations and physical corrections". In: *IEEE Transactions on Robotics* 36.3 (2020), pp. 835–854.

[35] Léon Bottou. "Online learning and stochastic approximations". In: *On-line Learning in Neural Networks*. Cambridge Univ Press, 1998, pp. 9–42.

[36] Oliver Brock and Oussama Khatib. "Elastic strips: A framework for motion generation in human environments". In: *The International Journal of Robotics Research* 21.12 (2002), pp. 1031–1052.

[37] Daniel S Brown and Scott Niekum. "Toward Probabilistic Safety Bounds for Robot Learning from Demonstration." In: 2017.

[38] Daniel S. Brown, Yuchen Cui, and Scott Niekum. "Risk-Aware Active Inverse Reinforcement Learning". In: *Proceedings of The 2nd Conference on Robot Learning*. Ed. by Aude Billard et al. Vol. 87. Proceedings of Machine Learning Research. PMLR, 29–31 Oct 2018, pp. 362–372. URL: http://proceedings.mlr.press/v87/brown18a.html.

[39] Maya Cakmak and Manuel Lopes. "Algorithmic and Human Teaching of Sequential Decision Tasks". In: *AAAI*. 2012, pp. 1536–1542.

[40] Eduardo F Camacho and Carlos Bordons Alba. *Model predictive control*. Springer science & business media, 2013.

[41] Peter Carruthers and Peter K Smith. *Theories of theories of mind*. Cambridge university press, 1996.

[42] Sergio Casas, Wenjie Luo, and Raquel Urtasun. "Intentnet: Learning to predict intention from raw sensor data". In: *CoRL*. 2018, pp. 947–956.

[43] Carlos Celemin, Javier Ruiz-del-Solar, and Jens Kober. "A fast hybrid reinforcement learning framework with human corrective feedback". In: *Autonomous Robots* 43.5 (June 2019), pp. 1173–1186. ISSN: 1573-7527. DOI: 10.1007/s10514-018-9786-6. URL: https://doi.org/10.1007/s10514-018-9786-6.

[44] Yuning Chai et al. "Multipath: Multiple probabilistic anchor trajectory hypotheses for behavior prediction". In: *arXiv preprint arXiv:1910.05449* (2019).

[45] Georgios C Chasparis and Jeff S Shamma. "Linear-programming-based multi-vehicle path planning with adversaries". In: *American Control Conference, 2005. Proceedings of the 2005*. IEEE. 2005, pp. 1072–1077.

[46] Min Chen et al. "Planning with Trust for Human-Robot Collaboration". In: *ACM/IEEE International Conference on Human-Robot Interaction (HRI)*. 2018, pp. 307–315.

[47] Mo Chen, Sylvia Herbert, and Claire J Tomlin. "Fast reachable set approximations via state decoupling disturbances". In: *Decision and Control (CDC), 2016 IEEE 55th Conference on*. IEEE. 2016, pp. 191–196.

[48] Mo Chen, Jennifer C Shih, and Claire J Tomlin. "Multi-vehicle collision avoidance via hamilton-jacobi reachability and mixed integer programming". In: *Decision and Control (CDC), 2016 IEEE 55th Conference on*. IEEE. 2016, pp. 1695–1700.

[49] Mo Chen et al. "A General System Decomposition Method for Computing Reachable Sets and Tubes". In: *IEEE Transactions on Automatic Control (to appear)* (2016).

[50] Mo Chen et al. "Decomposition of reachable sets and tubes for a class of nonlinear systems". In: *IEEE Transactions on Automatic Control* (2018).

[51] Mo Chen et al. "Safe platooning of unmanned aerial vehicles via reachability". In: *2015 54th IEEE conference on decision and control (CDC)*. IEEE. 2015, pp. 4695–4701.

[52] Xin Chen, Erika Ábrahám, and Sriram Sankaranarayanan. "Flow*: An analyzer for non-linear hybrid systems". In: *CAV*. 2013, pp. 258–263.

[53] Yujiao Cheng et al. "Human motion prediction using semi-adaptable neural networks". In: *2019 American Control Conference (ACC)*. IEEE. 2019, pp. 4884–4890.

[54] Jaedeug Choi and Kee-Eung Kim. "MAP inference for Bayesian inverse reinforcement learning". In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2011, pp. 1989–1997.

[55] Paul Christiano et al. "Deep reinforcement learning from human preferences". In: (June 2017).

[56] Yao-Li Chuang et al. "Multi-vehicle flocking: scalability of cooperative control algorithms using pairwise potentials". In: *Robotics and Automation, 2007 IEEE International Conference on*. IEEE. 2007, pp. 2292–2299.

[57] Michael G Crandall and Pierre-Louis Lions. "Viscosity solutions of Hamilton-Jacobi equations". In: *Transactions of the American mathematical society* 277.1 (1983), pp. 1–42.

[58] Alessandro De Luca et al. "Collision detection and safe reaction with the DLR-III lightweight manipulator arm". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2006, pp. 1623–1630.

[59] Agostino De Santis et al. "An atlas of physical human–robot interaction". In: *Mechanism and Machine Theory* 43.3 (2008), pp. 253–270.

[60] Ankush Desai et al. "Drona: A framework for safe distributed mobile robotics". In: *Proceedings of the 8th International Conference on Cyber-Physical Systems*. ACM. 2017, pp. 239–248.

[61] Aparna Dhinakaran et al. "A hybrid framework for multi-vehicle collision avoidance". In: *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*. IEEE. 2017, pp. 2979–2984.

[62] Hao Ding et al. "Human arm motion modeling and long-term prediction for safe and efficient human-robot-interaction". In: *Robotics and Automation (ICRA), 2011 IEEE International Conference on*. IEEE. 2011, pp. 5875–5880.

[63] Badis Djeridane and John Lygeros. "Neural approximation of PDE solutions: An application to reachability computations". In: *Proceedings of the 45th IEEE Conference on Decision and Control*. IEEE. 2006, pp. 3034–3039.

[64] Anca Dragan and Siddhartha Srinivasa. "Generating legible motion". In: (2013).

[65] Anca D Dragan and Siddhartha S Srinivasa. "A policy-blending formalism for shared control". In: *The International Journal of Robotics Research* 32.7 (2013), pp. 790–805.

[66] Anca D Dragan et al. "Movement primitives via optimization". In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2015, pp. 2339–2346.

[67] Katherine Driggs-Campbell, Roy Dong, and Ruzena Bajcsy. "Robust, Informative Human-in-the-Loop Predictions via Empirical Reachable Sets". In: *IEEE Transactions on Intelligent Vehicles* (2018).

[68] Robert James Elliott and Nigel John Kalton. *The existence of value in differential games*. Vol. 126. American Mathematical Soc., 1972.

[69] L. C. Evans and P. E. Souganidis. “Differential games and representation formulas for solutions of Hamilton-Jacobi-Isaacs equations”. In: *Indiana University mathematics journal* 33.5 (1984), pp. 773–797.

[70] Chelsea Finn, Sergey Levine, and Pieter Abbeel. “Guided cost learning: Deep inverse optimal control via policy optimization”. In: *International Conference on Machine Learning (ICML)*. 2016, pp. 49–58.

[71] Paolo Fiorini and Zvi Shiller. “Motion planning in dynamic environments using velocity obstacles”. In: *The International Journal of Robotics Research* 17.7 (1998), pp. 760–772.

[72] Jaime F Fisac et al. “A general safety framework for learning-based control in uncertain robotic systems”. In: *IEEE Transactions on Automatic Control* 64.7 (2018), pp. 2737–2752.

[73] Jaime F Fisac et al. “Hierarchical Game-Theoretic Planning for Autonomous Vehicles”. In: *arXiv preprint arXiv:1810.05766* (2018).

[74] Jaime F Fisac et al. “Probabilistically Safe Robot Planning with Confidence-Based Human Predictions”. In: *Robotics: Science and Systems (RSS)* (2018).

[75] Jaime F Fisac et al. “Reach-avoid problems with time-varying dynamics, targets and constraints”. In: *Proceedings of the 18th international conference on hybrid systems: computation and control*. 2015, pp. 11–20.

[76] Jaime F. Fisac et al. “Pragmatic-Pedagogic Value Alignment”. In: *CoRR* abs/1707.06354 (2017).

[77] David Fridovich-Keil et al. “Confidence-aware motion prediction for real-time collision avoidance”. In: *International Journal of Robotics Research* (2019).

[78] David Fridovich-Keil et al. “Planning, Fast and Slow: A Framework for Adaptive Real-Time Safe Trajectory Planning.” In: *IEEE Conference on Robotics and Automation* (2018).

[79] Jie Fu and Ufuk Topcu. “Pareto efficiency in synthesizing shared autonomy policies with temporal logic constraints”. In: *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2015, pp. 361–368.

[80] Justin Fu et al. “Variational inverse control with events: A general framework for data-driven reward definition”. In: *Advances in neural information processing systems* 31 (2018).

[81] György Gergely and Gergely Csibra. “Teleological reasoning in infancy: The naıve theory of rational action”. In: *Trends in cognitive sciences* 7.7 (2003), pp. 287–292.

[82] Antoine Girard. “Reachability of uncertain linear systems using zonotopes”. In: *International Workshop on Hybrid Systems: Computation and Control*. Springer. 2005, pp. 291–305.

[83] Sally A Goldman and Michael J Kearns. "On the complexity of teaching". In: *Journal of Computer and System Sciences* 50.1 (1995), pp. 20–31.

[84] Michael A Goodrich. "Potential fields tutorial". In: *Class Notes* 157 (2002).

[85] Reymundo Gutierrez et al. "Incremental Task Modification via Corrective Demonstrations". In: *2018 IEEE International Conference on Robotics and Automation (ICRA)* (2018), pp. 1126–1133.

[86] Isabelle Guyon and André Elisseeff. "An introduction to variable and feature selection". In: *Journal of Machine Learning Research* 3 (2003), pp. 1157–1182.

[87] Sami Haddadin and Elizabeth Croft. "Physical human–robot interaction". In: *Springer Handbook of Robotics*. Springer, 2016, pp. 1835–1874.

[88] Sami Haddadin et al. "Collision detection and reaction: A contribution to safe physical human-robot interaction". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2008, pp. 3356–3363.

[89] Dylan Hadfield-Menell et al. "Cooperative inverse reinforcement learning". In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2016, pp. 3909–3917.

[90] Dylan Hadfield-Menell et al. "Inverse Reward Design". In: *NIPS*. 2017.

[91] Jason Hardy and Mark Campbell. "Contingency planning over probabilistic obstacle predictions for autonomous road vehicles". In: *IEEE Transactions on Robotics* 29.4 (2013), pp. 913–929.

[92] Peter E Hart, Nils J Nilsson, and Bertram Raphael. "A formal basis for the heuristic determination of minimum cost paths". In: *IEEE transactions on Systems Science and Cybernetics* 4.2 (1968), pp. 100–107.

[93] Alain Haurie. "Feedback equilibria in differential games with structural and modal uncertainties". In: *in Advances in Large Scale Systems*. Citeseer. 1984.

[94] Kelsey P Hawkins et al. "Probabilistic human action prediction and wait-sensitive planning for responsive human-robot collaboration". In: *Humanoid Robots (Humanoids), 2013 13th IEEE-RAS International Conference on*. IEEE. 2013, pp. 499–506.

[95] Sylvia L. Herbert et al. "FaSTrack: a Modular Framework for Fast and Guaranteed Safe Motion Planning". In: *IEEE Conference on Decision and Control* (2017).

[96] Guy Hoffman and Cynthia Breazeal. "Cost-based anticipatory action selection for human–robot fluency". In: *IEEE Transactions on Robotics* 23.5 (2007), pp. 952–961.

[97] Neville Hogan. "Impedance control: An approach to manipulation; Part II—Implementation". In: *Journal of Dynamic Systems, Measurement, and Control* 107.1 (1985), pp. 8–16.

[98] Haomiao Huang et al. "A differential game approach to planning in adversarial scenarios: A case study on capture-the-flag". In: *2011 IEEE International Conference on Robotics and Automation*. IEEE. 2011, pp. 1451–1456.

[99] Ashesh Jain et al. “Learning preferences for manipulation tasks from online coactive feedback”. In: *The International Journal of Robotics Research* 34.10 (2015), pp. 1296–1313.

[100] Siddarth Jain and Brenna Argall. “Recursive Bayesian human intent recognition in shared-control robotics”. In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2018, pp. 3905–3912.

[101] Nathanaël Jarrassé, Themistoklis Charalambous, and Etienne Burdet. “A framework to describe, analyze and generate interactive motor behaviors”. In: *PloS ONE* 7.11 (2012), e49945.

[102] Shervin Javdani et al. “Shared autonomy via hindsight optimization for teleoperation and teaming”. In: *The International Journal of Robotics Research* 37.7 (2018), pp. 717–742.

[103] E. T. Jaynes. “Information Theory and Statistical Mechanics”. In: *Phys. Rev.* 106 (4 May 1957), pp. 620–630. DOI: 10.1103/PhysRev.106.620. URL: https://link.aps.org/doi/10.1103/PhysRev.106.620.

[104] Hong J Jeon and Anca D Dragan. “Configuration Space Metrics”. In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2018, pp. 5101–5108.

[105] Frank Jiang et al. “Using neural networks to compute approximate and guaranteed feasible Hamilton-Jacobi-Bellman PDE solutions”. In: *arXiv preprint arXiv:1611.03158* (2016).

[106] Ananth Jonnavittula and Dylan P Losey. “I know what you meant: Learning human objectives by (under)estimating their choice set”. In: *IEEE International Conference on Robotics and Automation (ICRA)* (2021).

[107] Leslie Pack Kaelbling, Michael L Littman, and Anthony R Cassandra. “Planning and acting in partially observable stochastic domains”. In: *Artificial Intelligence* 101.1-2 (1998), pp. 99–134.

[108] Mrinal Kalakrishnan et al. “Learning objective functions for manipulation”. In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2013, pp. 1331–1336.

[109] Rudolf Emil Kalman. “When is a linear control system optimal?” In: *Journal of Basic Engineering* 86.1 (1964), pp. 51–60.

[110] Sertac Karaman and Emilio Frazzoli. “Sampling-based algorithms for optimal motion planning”. In: *The international journal of robotics research* 30.7 (2011), pp. 846–894.

[111] Sertac Karaman and Emilio Frazzoli. “Sampling-based algorithms for optimal motion planning”. In: *The International Journal of Robotics Research* 30.7 (2011), pp. 846–894.

[112] Vasiliy Karasev et al. “Intent-aware long-term prediction of pedestrian motion”. In: *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2016, pp. 2543–2549.

[113] Martin Karlsson, Anders Robertsson, and Rolf Johansson. “Autonomous Interpretation of Demonstrations for Modification of Dynamical Movement Primitives”. In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2017, pp. 316–321.

[114] D.F. Keil, J. Fisac, and C. J. Tomlin. “Safe and Complete Real-Time Planning and Exploration in Unknown Environments”. In: *arXiv preprint* (2018).

[115] Mahdi Khoramshahi and Aude Billard. “A dynamical system approach to task-adaptation in physical human–robot interaction”. In: *Autonomous Robots* 43.4 (2019), pp. 927–946.

[116] Mahdi Khoramshahi et al. “From human physical interaction to online motion adaptation using parameterized dynamical systems”. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2018, pp. 1361–1366.

[117] Kris M Kitani et al. “Activity forecasting”. In: *ECCV*. 2012, pp. 201–214.

[118] Ross A Knepper and Daniela Rus. “Pedestrian-inspired sampling-based multi-robot collision avoidance”. In: *RO-MAN, 2012 IEEE*. IEEE. 2012, pp. 94–100.

[119] Niklas Kochdumper and Matthias Althoff. “Sparse polynomial zonotopes: A novel set representation for reachability analysis”. In: *IEEE Transactions on Automatic Control* 66.9 (2020), pp. 4043–4058.

[120] Mykel J Kochenderfer et al. “Airspace encounter models for estimating collision risk”. In: *Journal of Guidance, Control, and Dynamics* 33.2 (2010), pp. 487–499.

[121] J Zico Kolter et al. “A probabilistic approach to mixed open-loop and closed-loop control, with application to extreme autonomous driving”. In: *2010 IEEE International Conference on Robotics and Automation*. IEEE. 2010, pp. 839–845.

[122] Florian Köpf et al. “Inverse Reinforcement Learning for Identification in Linear-Quadratic Dynamic Games”. In: *IFAC-PapersOnLine* 50.1 (2017). 20th IFAC World Congress, pp. 14902–14908. ISSN: 2405-8963. DOI: https://doi.org/10.1016/j.ifacol.2017.08.2537. URL: http://www.sciencedirect.com/science/article/pii/S2405896317334596.

[123] Hema Swetha Koppula and Ashutosh Saxena. “Anticipating human activities for reactive robotic response.” In: *IROS*. 2013, p. 2071.

[124] S. Kousik et al. “Bridging the Gap Between Safety and Real-Time Performance in Receding-Horizon Trajectory Design for Mobile Robots”. In: *arXiv preprint* (2018).

[125] Donald H. Kraft. “A software package for sequential quadratic programming”. In: 1988.

[126] Thibault Kruse et al. “Human-aware robot navigation: A survey”. In: *Robotics and Autonomous Systems* 61.12 (2013), pp. 1726–1743.

[127] Tomasz Piotr Kucner et al. "Enabling flow awareness for mobile robots in partially observable environments". In: *IEEE Robotics and Automation Letters* 2.2 (2017), pp. 1093–1100.

[128] Markus Kuderer et al. "Feature-Based Prediction of Trajectories for Socially Compliant Navigation." In: *Robotics: science and systems*. 2012.

[129] Maxime Laborde and Adam Oberman. "A Lyapunov analysis for accelerated gradient methods: From deterministic to stochastic case". In: *International Conference on Artificial Intelligence and Statistics*. 2020, pp. 602–612.

[130] M. Lahijanian et al. "Iterative Temporal Planning in Uncertain Environments with Partial Satisfaction Guarantees". In: *IEEE Trans. on Robotics* 32 (2016), pp. 583–599.

[131] Benoit Landry et al. "Reach-avoid problems via sum-or-squares optimization and dynamic programming". In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2018, pp. 4325–4332.

[132] Przemyslaw A Lasota and Julie A Shah. "A multiple-predictor approach to human motion prediction". In: *ICRA*. 2017, pp. 2300–2307.

[133] Przemyslaw A Lasota and Julie A Shah. "Analyzing the effects of human-aware motion planning on close-proximity human–robot collaboration". In: *Human factors* 57.1 (2015), pp. 21–33.

[134] Karen Leung et al. "On infusing reachability-based safety assurance within planning frameworks for human–robot vehicle interactions". In: *The International Journal of Robotics Research* 39.10-11 (2020), pp. 1326–1345.

[135] Sergey Levine and Vladlen Koltun. "Continuous Inverse Optimal Control with Locally Optimal Examples". In: *ArXiv* abs/1206.4617 (2012).

[136] Sergey Levine et al. "End-to-end training of deep visuomotor policies". In: *Journal of Machine Learning Research* 17.1 (2016), pp. 1334–1373.

[137] Frank L Lewis et al. *Cooperative control of multi-agent systems: optimal and adaptive design approaches*. Springer Science & Business Media, 2013.

[138] Anjian Li et al. "Prediction-Based Reachability for Collision Avoidance in Autonomous Driving". In: *2021 IEEE International Conference on Robotics and Automation (ICRA)* (2021).

[139] Mengxi Li et al. "Learning Human Objectives from Sequences of Physical Corrections". In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2021.

[140] Yanan Li et al. "A framework of human–robot coordination based on game theory and policy iteration". In: *IEEE Transactions on Robotics* 32.6 (2016), pp. 1408–1418.

[141] Feng-Li Lian and Richard Murray. "Real-time trajectory generation for the cooperative path planning of multi-vehicle systems". In: *Decision and Control, 2002, Proceedings of the 41st IEEE Conference on*. Vol. 4. IEEE. 2002, pp. 3766–3769.

[142] Michael L Littman, Anthony R Cassandra, and Leslie Pack Kaelbling. "Learning policies for partially observable environments: Scaling up". In: *International Conference on Machine Learning (ICML)*. 1995, pp. 362–370.

[143] Stefan B Liu et al. "Provably safe motion of mobile robots in human environments". In: *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2017, pp. 1351–1357.

[144] Dylan P Losey and Marcia K O'Malley. "Including uncertainty when learning from human corrections". In: *arXiv preprint arXiv:1806.02454* (2018).

[145] Dylan P Losey and Marcia K O'Malley. "Learning the correct robot trajectory in real-time from physical human interactions". In: *ACM Transactions on Human-Robot Interaction* (2020).

[146] Dylan P Losey and Marcia K O'Malley. "Trajectory deformations from physical human–robot interaction". In: *IEEE Transactions on Robotics* 34.1 (2018), pp. 126–138.

[147] Dylan P Losey et al. "A review of intent detection, arbitration, and communication aspects of shared control for physical human–robot interaction". In: *Applied Mechanics Reviews* 70.1 (2018).

[148] Dylan P Losey et al. "Physical interaction as communication: Learning robot objectives online from human corrections". In: *The International Journal of Robotics Research* (2021), p. 02783649211050958.

[149] R. Duncan Luce. *Individual choice behavior: A theoretical analysis*. Wiley, 1959.

[150] John Lygeros. "On reachability and minimum cost optimal control". In: *Automatica*. Vol. 40. 6. Elsevier, 2004, pp. 917–927.

[151] Wei-Chiu Ma et al. "Forecasting Interactive Dynamics of Pedestrians With Fictitious Play". In: *CVPR*. 2017.

[152] David JC MacKay, David JC Mac Kay, et al. *Information theory, inference and learning algorithms*. Cambridge university press, 2003.

[153] Jim Mainprice and Dmitry Berenson. "Human–robot collaborative manipulation planning using early prediction of human motion". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2013, pp. 299–306.

[154] Anirudha Majumdar and Russ Tedrake. "Funnel libraries for real-time robust feedback motion planning". In: *Int. J. Robotics Research* (June 2017). DOI: 10.1177/0278364917712421.

[155] Kostas Margellos and John Lygeros. "Hamilton–Jacobi formulation for reach–avoid differential games". In: *IEEE Transactions on automatic control* 56.8 (2011), pp. 1849–1861.

[156] José Ramón Medina, Tamara Lorenz, and Sandra Hirche. "Synthesizing anticipatory haptic assistance considering human behavior uncertainty". In: *IEEE Transactions on Robotics* 31.1 (2015), pp. 180–190.

[157] Ian Mitchell. *A Toolbox of Level Set Methods*. `http://people.cs.ubc.ca/~mitchell/ToolboxLS/index.html`. 2009.

[158] Ian M. Mitchell, A. M. Bayen, and C. J. Tomlin. "A time-dependent Hamilton-Jacobi formulation of reachable sets for continuous dynamic games". In: *IEEE Transactions on Automatic Control* 50.7 (2005), pp. 947–957. DOI: `10.1109/TAC.2005.851439`. URL: `http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1463302`.

[159] John B Moore and Uwe Helmke. "Optimization and Dynamical Systems". In: (1996).

[160] Shingo Murata et al. "Achieving Human–Robot Collaboration with Dynamic Goal Inference by Gradient Descent". In: *International Conference on Neural Information Processing*. Springer. 2019, pp. 579–590.

[161] Thulasi Mylvaganam, Mario Sassano, and Alessandro Astolfi. "A differential game approach to multi-agent collision avoidance". In: *IEEE Transactions on Automatic Control* 62.8 (2017), pp. 4229–4235.

[162] Andrew Y Ng and Stuart J Russell. "Algorithms for inverse reinforcement learning". In: *International Conference on Machine Learning (ICML)*. 2000, pp. 663–670.

[163] KN Niarchos and John Lygeros. "A neural approximation to continuous time reachability computations". In: *Proceedings of the 45th IEEE Conference on Decision and Control*. IEEE. 2006, pp. 6313–6318.

[164] Stefanos Nikolaidis et al. "Game-theoretic modeling of human adaptation in human-robot collaboration". In: *Proceedings of the 2017 ACM/IEEE international conference on human-robot interaction*. 2017, pp. 323–331.

[165] Stefanos Nikolaidis et al. "Human-robot mutual adaptation in shared autonomy". In: *ACM/IEEE International Conference on Human-Robot Interaction (HRI)*. 2017, pp. 294–302.

[166] Reza Olfati-Saber and Richard M Murray. "Distributed cooperative control of multiple vehicle formations using structural potential functions". In: *IFAC world congress*. Vol. 15. Citeseer. 2002, pp. 242–248.

[167] Takayuki Osa et al. "An Algorithmic Perspective on Imitation Learning". In: *Foundations and Trends in Robotics* 7.1-2 (2018), pp. 1–179.

[168] Lev Semenovich Pontryagin. *Mathematical theory of optimal processes*. CRC press, 1987.

[169] Morgan Quigley et al. "ROS: an Open-Source Robot Operating System". In: *ICRA Workshop on Open Source Software*. 2009.

[170] Sean Quinlan and Oussama Khatib. “Elastic bands: Connecting path planning and control”. In: *IEEE International Conference on Robotics and Automation (ICRA)*. 1993, pp. 802–807.

[171] Rajesh Rajamani. *Vehicle dynamics and control*. Springer Science & Business Media, 2011.

[172] Saša V Raković. “Set theoretic methods in model predictive control”. In: *Nonlinear Model Predictive Control*. Springer, 2009, pp. 41–54.

[173] Deepak Ramachandran and Eyal Amir. “Bayesian inverse reinforcement learning”. In: *Urbana* 51.61801 (2007), pp. 1–4.

[174] Amir Rasouli et al. “PIE: A large-scale dataset and models for pedestrian intention estimation and trajectory prediction”. In: *ICCV*. 2019, pp. 6262–6271.

[175] Nathan D Ratliff, J Andrew Bagnell, and Martin A Zinkevich. “Maximum margin planning”. In: *International Conference on Machine Learning (ICML)*. 2006, pp. 729–736.

[176] Ellis Ratner et al. “Efficient Dynamics Estimation With Adaptive Model Sets”. In: *IEEE Robotics and Automation Letters* 6.2 (2021), pp. 2373–2380.

[177] Harish Chaandar Ravichandar and Ashwin Dani. “Human intention inference and motion modeling using approximate em with online learning”. In: *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, pp. 1819–1824.

[178] Christoph Rösmann et al. “Online trajectory prediction and planning for social robot navigation”. In: *AIM*. 2017.

[179] Vicenc Rubies Royo et al. “Classification-based Approximate Reachability with Guarantees Applied to Safe Trajectory Tracking”. In: *arXiv preprint arXiv:1803.03237* (2018).

[180] Wenjie Ruan, Xiaowei Huang, and Marta Kwiatkowska. “Reachability analysis of deep neural networks with provable guarantees”. In: *arXiv preprint arXiv:1805.02242* (2018).

[181] Vicenç Rubies-Royo and Claire Tomlin. “Recursive regression with neural networks: Approximating the hji pde solution”. In: *arXiv preprint arXiv:1611.02739* (2016).

[182] Andrey Rudenko, Luigi Palmieri, and Kai O Arras. “Predictive planning for a mobile robot in human environments”. In: *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA), Works. on AI Planning and Robotics*. 2017.

[183] Andrey Rudenko et al. “Human Motion Trajectory Prediction: A Survey”. In: *IJRR*. 2019.

[184] Dorsa Sadigh et al. “Active Preference-Based Learning of Reward Functions”. In: *Robotics: Science and Systems*. 2017.

[185] Dorsa Sadigh et al. "Data-driven probabilistic modeling and verification of human driver behavior". In: *AAAI Spring Symposium-Technical Report*. 2014, pp. 56–61.

[186] Dorsa Sadigh et al. "Information gathering actions over human internal state". In: *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2016, pp. 66–73.

[187] Dorsa Sadigh et al. "Planning for Autonomous Cars that Leverage Effects on Human Actions". In: *Robotics: Science and Systems (RSS)* (2016).

[188] Tim Salzmann et al. "Trajectron++: Dynamically-feasible trajectory forecasting with heterogeneous data". In: *European Conference on Computer Vision*. Springer. 2020, pp. 683–700.

[189] S. Sarid, Bingxin X., and H. Kress-gazit. "Guaranteeing high-level behaviors while exploring partially known maps". In: *RSS*. 2012.

[190] Edward Schmerling et al. "Multimodal Probabilistic Model-Based Planning for Human-Robot Interaction". In: *arXiv preprint arXiv:1710.09483* (2017).

[191] Friederike Schneemann and Patrick Heinemann. "Context-based detection of pedestrian crossing intention for autonomous driving in urban environments". In: *IROS*. 2016.

[192] John Schulman et al. "Motion planning with sequential convex optimization and convex collision checking". In: *The International Journal of Robotics Research* 33.9 (2014), pp. 1251–1270.

[193] Wilko Schwarting, Javier Alonso-Mora, and Daniela Rus. "Planning and decision-making for autonomous vehicles". In: *Annual Review of Control, Robotics, and Autonomous Systems* 1 (2018), pp. 187–210.

[194] Wilko Schwarting et al. "Social behavior for autonomous vehicles". In: *Proceedings of the National Academy of Sciences* 116.50 (2019), pp. 24972–24978.

[195] Bin Shi et al. "Understanding the acceleration phenomenon via high-resolution differential equations". In: *arXiv preprint arXiv:1810.08907* (2018).

[196] Jennifer C Shih. "Predicting Stochastic Human Forward Reachable Sets Based on Learned Human Behavior". In: *2019 American Control Conference (ACC)*. IEEE. 2019, pp. 5247–5253.

[197] Pannaga Shivaswamy and Thorsten Joachims. "Coactive Learning". In: *Journal of Artificial Intelligence Research* 53 (2015), pp. 1–40.

[198] David Silver and Joel Veness. "Monte-Carlo planning in Large POMDPs". In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2010, pp. 2164–2172.

[199] Sumeet Singh et al. "Robust online motion planning via contraction theory and convex optimization". In: *Proc. IEEE Int. Conf. Robotics and Automation*. 2017.

[200] Sumeet Singh et al. “Robust Tracking with Model Mismatch for Fast and Safe Planning: an SOS Optimization Approach”. In: *arXiv preprint arXiv:1808.00649* (2018).

[201] Emrah Akin Sisbot et al. “A human aware mobile robot motion planner”. In: *IEEE Transactions on Robotics* 23.5 (2007), pp. 874–883.

[202] Beate Sodian, Barbara Schoeppner, and Ulrike Metz. “Do infants apply the principle of rational action to human agents?” In: *Infant Behavior and Development* 27.1 (2004), pp. 31–41.

[203] Adhiraj Somani et al. “DESPOT: Online POMDP planning with regularization”. In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2013, pp. 1772–1780.

[204] Mark W Spong, Seth Hutchinson, and Mathukumalli Vidyasagar. *Robot Modeling and Control*. Vol. 3. New York, NY, USA: John Wiley & Sons, 2006.

[205] Liting Sun, Xiaogang Jia, and Anca D Dragan. “On complementing end-to-end human motion predictors with planning”. In: *2021 Robotics: Science and Systems (RSS)*. 2021.

[206] Zachary Sunberg and Mykel Kochenderfer. “POMCPOW: An online algorithm for POMDPs with continuous state, action, and observation spaces”. In: *arXiv preprint arXiv:1709.06196* (2017).

[207] Andrea L Thomaz and Cynthia Breazeal. “Teachable robots: Understanding human teaching behavior to build more effective robot learners”. In: *Artificial Intelligence* 172.6-7 (2008), pp. 716–737.

[208] Andrea L Thomaz and Maya Cakmak. “Learning about objects with human teachers”. In: *ACM/IEEE International Conference on Human-Robot Interaction (HRI)*. 2009, pp. 15–22.

[209] Ran Tian, Liting Sun, and Masayoshi Tomizuka. “Bounded risk-sensitive markov games: Forward policy design and inverse reward learning with iterative reasoning and cumulative prospect theory”. In: *AAAI Conference on Artificial Intelligence*. 2021.

[210] Ran Tian et al. “Anytime Game-Theoretic Planning with Active Reasoning About Humans’ Latent States for Human-Centered Robots”. In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2021.

[211] Ran Tian et al. “Safety Assurances for Human-Robot Interaction via Confidence-aware Game-theoretic Human Models”. In: *International Conference on Robotics and Automation (ICRA)* (2022).

[212] Ekaterina Tolstaya et al. “Identifying driver interactions via conditional behavior prediction”. In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2021, pp. 3473–3479.

[213] Alejandro Torreño et al. “Cooperative multi-agent planning: a survey”. In: *ACM Computing Surveys (CSUR)* 50.6 (2017), p. 84.

[214] Peter Trautman and Andreas Krause. “Unfreezing the robot: Navigation in dense, interacting crowds”. In: *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*. IEEE. 2010, pp. 797–803.

[215] Jur Van den Berg, Ming Lin, and Dinesh Manocha. “Reciprocal velocity obstacles for real-time multi-agent navigation”. In: *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*. IEEE. 2008, pp. 1928–1935.

[216] Pravin P Varaiya. “On the existence of solutions to a differential game”. In: *SIAM Journal on Control* 5.1 (1967), pp. 153–162.

[217] John Von Neumann and Oskar Morgenstern. *Theory of games and economic behavior*. Princeton University Press Princeton, NJ, 1945.

[218] Heinrich Von Stackelberg. *Market structure and equilibrium*. Springer Science & Business Media, 2010.

[219] R. Walambe et al. “Optimal trajectory generation for car-type mobile robot using spline interpolation”. In: *IFAC*. 1. 2016, pp. 601–606.

[220] Zijian Wang, Riccardo Spica, and Mac Schwager. “Game Theoretic Motion Planning for Multi-Robot Racing”. In: (2018).

[221] Kevin Waugh, Brian D Ziebart, and J Andrew Bagnell. “Inverse Correlated Equilibrium for Matrix Games”. In: *Advances in Neural Information Processing Systems (NIPS)* (2010).

[222] Ashia C Wilson, Benjamin Recht, and Michael I Jordan. “A lyapunov analysis of momentum methods in optimization”. In: *Journal of Machine Learning Research* (2021).

[223] Albert Wu and Jonathan P How. “Guaranteed infinite horizon avoidance of unpredictable, dynamically constrained obstacles”. In: *Autonomous robots* 32.3 (2012), pp. 227–242.

[224] Zheng Wu et al. “Efficient sampling-based maximum entropy inverse reinforcement learning with application to autonomous driving”. In: *IEEE Robotics and Automation Letters* 5.4 (2020), pp. 5355–5362.

[225] Markus Wulfmeier, Peter Ondruska, and Ingmar Posner. “Maximum entropy deep inverse reinforcement learning”. In: *arXiv preprint arXiv:1507.04888* (2015).

[226] Insoon Yang et al. “One-shot computation of reachable sets for differential games”. In: *Proceedings of the 16th international conference on Hybrid systems: computation and control*. 2013, pp. 183–192.

[227] Liren Yang and Necmiye Ozay. “Scalable zonotopic under-approximation of backward reachable sets for uncertain linear systems”. In: *IEEE Control Systems Letters* 6 (2021), pp. 1555–1560.

[228] Hang Yin et al. “An ensemble inverse optimal control approach for robotic task learning and adaptation”. In: *Autonomous Robots* 43.4 (2019), pp. 875–896.

[229] Tianhe Yu et al. "One-shot imitation from observing humans via domain-adaptive meta-learning". In: *arXiv preprint arXiv:1802.01557* (2018).

[230] Wei Zhan et al. "INTERACTION Dataset: An INTERnational, Adversarial and Cooperative moTION Dataset in Interactive Driving Scenarios with Semantic Maps". In: *arXiv:1910.03088 [cs, eess]* (Sept. 2019).

[231] Zixu Zhang and Jaime F Fisac. "Safe Occlusion-aware Autonomous Driving via Game-Theoretic Active Perception". In: *arXiv preprint arXiv:2105.08169* (2021).

[232] Hang Zhao et al. "Tnt: Target-driven trajectory prediction". In: *arXiv preprint arXiv:2008.08294* (2020).

[233] Jiangchuan Zheng, Siyuan Liu, and Lionel M. Ni. "Robust Bayesian Inverse Reinforcement Learning with Sparse Behavior Noise". In: *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*. AAAI'14. AAAI Press, 2014.

[234] Dingjiang Zhou, Zijian Wang, and Mac Schwager. "Agile Coordination and Assistive Collision Avoidance for Quadrotor Swarms Using Virtual Structures". In: *IEEE Transactions on Robotics* 34.4 (2018), pp. 916–923.

[235] Dingjiang Zhou et al. "Fast, On-line Collision Avoidance for Dynamic Vehicles using Buffered Voronoi Cells". In: *IEEE Robotics and Automation Letters (RA-L)* 2 (2017), pp. 1047–1054. DOI: 10.1109/LRA.2017.2656241.

[236] Xiaojin Zhu. "Machine Teaching: An Inverse Problem to Machine Learning and an Approach Toward Optimal Education". In: *AAAI*. 2015, pp. 4083–4087.

[237] Brian D Ziebart et al. "Maximum entropy inverse reinforcement learning." In: *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence*. Vol. 8. Chicago, IL, USA. 2008, pp. 1433–1438.

[238] Brian D. Ziebart et al. "Planning-based prediction for pedestrians". In: *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*. IEEE. 2009, pp. 3931–3936.