Optimizing for Robot Transparency

Sandy Huang

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2019-115
http://www2.eecs.berkeley.edu/Pubs/TechRpts/2019/EECS-2019-115.html

August 16, 2019

Optimizing for Robot Transparency

By

Sandy H. Huang

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Engineering – Electrical Engineering and Computer Sciences

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:
Professor Anca Dragan, Co-chair
Professor Pieter Abbeel, Co-chair
Professor Juliana Schroeder

Summer 2019

Optimizing for Robot Transparency

Abstract

Optimizing for Robot Transparency

by

Sandy H. Huang

Doctor of Philosophy in Engineering – Electrical Engineering and Computer Sciences

University of California, Berkeley

Professor Anca Dragan, Co-chair
Professor Pieter Abbeel, Co-chair

As robots become more capable and commonplace, it becomes increasingly important that they are *transparent* to humans. People need to have accurate mental models of a robot, so that they can anticipate what it will do, know when and where not to rely it, and understand why it failed. This helps engineers ensure safety and robustness of the robot systems they develop, and enables human end-users to interact more safely and seamlessly with robots.

This thesis introduces a framework for producing robot behavior that increases transparency. Our key insight is that a robot's actions do not just influence the physical world; they also inevitably influence a human observer's mental model of the robot. We attempt to model the latter—how humans might make inferences about a robot's objectives, policy, and capabilities from observations of its behavior—so that we can then present examples of robot behavior that optimally bring the human's understanding closer to the true robot model. In this way, our framework casts transparency as an optimization problem.

Part I introduces our framework of optimizing for robot transparency, and applies it in three ways: communicating a robot's objectives, which situations it can handle, and why it is incapable of performing a task. Part II investigates how transparency is useful not just for safe and seamless interaction, but also for learning. When humans teach a robot, giving human teachers transparency regarding what the robot has learned so far makes it easier for them to select informative teaching examples.

To my parents, Ningning Han and Xiaoqiu Huang

Contents

List of Figures

List of Tables

Acknowledgments

I am deeply grateful for the support I have received during my Ph.D. First, I would like to thank my co-advisors, Professors Anca Dragan and Pieter Abbeel. Both Anca and Pieter are amazing mentors, who care deeply about the success and wellbeing of their students. Anca introduced me to algorithmic HRI; my class project with her continued in several open-ended directions, and became the foundation of this thesis. I have learned so much from Anca and Pieter—how to pick impactful research problems, write clear papers, make pretty figures, give cohesive presentations, and mentor others in research. I am incredibly grateful for the time and effort they have dedicated to helping me grow.

I would also like to thank Professors Juliana Schroeder, John Canny, and Tom Griffiths, for serving on my qualifying and thesis committees, and for providing valuable feedback on the content of my thesis. I also learned a great deal from my peers at Berkeley, in particular from the other students in Anca and Pieter's lab. I want to especially thank those who contributed to the work in this thesis—Minae Kwon, Isabella Huang, Ravi Pandya, David Held, and Kush Bhatia.

I gratefully acknowledge the funding I received from the Berkeley Chancellor's Fellowship and the NSF Graduate Research Fellowship, which gave me the flexibility to explore and pursue my research interests, without being dependent on grant funding. Looking back, my undergraduate research experiences at Stanford were the initial steps that led me down the path to getting a Ph.D. I want to thank Professors Nigam Shah and Jure Leskovec for giving me the opportunity to learn in their labs.

Beyond research, I am grateful to my friends for helping me stay happy and maintain balance. I will always have fond memories of our Chinese New Year dumpling parties, fire trail runs, board game nights, and exciting post-conference travel. Thank you Hong Van Pham, Sophia Nguyen, Alex Lee, Robert Nishihara, Chelsea Finn, Dylan Hadfield-Menell, Carlos Florensa-Campo, Grant Ho, Richard Zhang, Jacob Andreas, Eriz Tzeng, John Bolander, Elliott Jin, and Adam Kosiorek.

Finally, I would like to thank my parents and family for their unwavering love and support. I couldn't have done this without you.

Chapter 1

Introduction

Interactions between robots and humans are becoming increasingly common and necessary, as robots become more capable and widely deployed. This is already happening in factories [1–3], hotels and hospitals [4, 5], homes [6, 7], and on the road [8]. Whether robots operate in collaboration with, side-by-side with, or in service of humans, making human-robot interactions safe and seamless for humans is essential.

Safe and seamless human-robot interaction depends on human end-users having a reasonably-accurate *mental model* of the robot: in other words, a robot's objectives, policy, and capabilities should be *transparent* to end-users. When humans have an inaccurate or incomplete mental model of a robot—and thus their expectations of the robot are mismatched with reality—there is a greater chance of negative outcomes. For instance, a safety driver may incorrectly expect an autonomous car to handle a safety-critical situation appropriately, and thus fail to take control from the car. Or, she may be surprised, scared, or panicked and unnecessarily take control when the car drives in an unexpected (but still safe) way.

Once human end-users start interacting with a robot in the real world, over time their mental model of the robot gradually improves; this is known as *familiarization* [9] (Fig. 1.1, gray). But this process of *passive* familiarization may take a while, because it depends on which situations the robot happens to encounter in the world. A passenger riding in an autonomous car might need hours, or even days of interaction in order to develop an intuitive understanding of how this car drives and which situations it can handle.

During passive familiarization, humans spend a significant amount of time interacting with robots that they have poor mental models of, which is dangerous. Another approach to familiarization, that works in highly structured environments like factories, is to enumerate how a robot will act in all possible situations. But

in most real-world domains in which we would like to deploy robots, this enumeration is impossible—for instance, the traffic situations that an autonomous car may encounter, or crowd configurations in a hotel lobby.

This thesis proposes an alternative to the existing options for familiarization. Our key insight is that a robot's actions not only affect the physical world, but also inevitably affect a human observer's mental model of that robot. With this in mind, we can optimize for *informative examples* of robot behavior to show human end-users, to speed up the familiarization process (Fig. 1.1, orange). "Informative" examples are those that optimally bring this mental model closer to the true robot model, given what we know (or assume) about how humans update their mental model of a robot based on observations of its behavior. In this way, our framework casts transparency as an optimization problem.

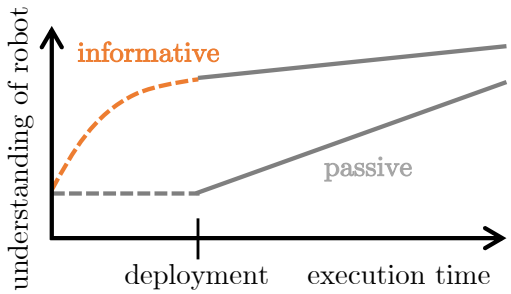


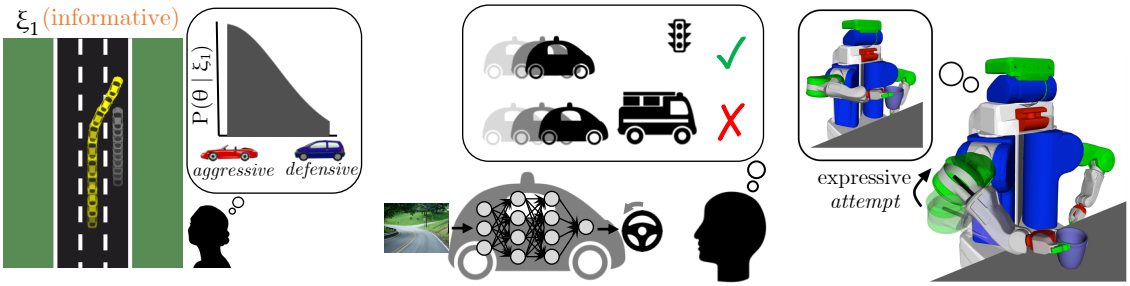
Figure 1.1: When human end-users see *informative* examples of a robot's behavior (orange), they more quickly improve their mental model of this robot, compared to the typical approach of relying on passive familiarization (gray).

Our framework can also be seen as optimizing for transparency via algorithmic teaching. In algorithmic teaching, the teacher knows how the student learns, and thus can select examples that optimally teach that student. In our framework, we have a model of how humans will make inferences about a robot's objectives, policy, and capabilities based on examples of robot behavior. Then we can leverage this model to select examples of robot behavior that optimally teach humans the robot's true objectives, policy, and capabilities.

Part I describes how we apply our framework toward achieving several possible aims of transparency:

- Chapter 3 tackles the question of how to help users anticipate how a robot will act, even in situations that they have not seen the robot act in before.[1] Our approach is to achieve this by giving users an intuitive understanding of the tradeoffs in a robot's objective function—in the case of autonomous cars, this encodes the car's driving style. In this setting, informative examples of robot behavior are those that help humans differentiate between different possible objectives. So, observing how an autonomous car drives on a mostly empty

[1] Chapter 3 is based on work published in RSS 2017 and AuRo 2019 [10, 11].



(a) (b) (c)

Figure 1.2: We investigate how showing informative examples of robot behavior can improve human end-users' understanding of (a) how a robot will act in new environments, (b) which situations a robot can handle, and (c) both what task a robot was trying to do and why it is incapable of accomplishing it.

road is not informative, because cars with different driving styles would still behave similarly; in contrast, observing how this autonomous car changes lanes when there is another car in the way is informative, because it highlights the tradeoff that this car makes between safety and efficiency (Fig. 1.2a). This work is related to a large body of prior work on inverse reinforcement learning [12], in which a robot learns an objective function from human demonstrations; in our case, we attempt to model how *humans* would infer a robot's objective from its behavior.

- Chapter 4 tackles a narrower goal, of communicating the robot's incapability with respect to a single task setup.[2] The hope is that if users understand both *what* a robot is trying to do and *why* it failed, they would be equipped to better assist it. We focus on robot failures that are caused by a dynamic constraint—e.g., trying to lift or push something that is too heavy, or trying to open a locked door or cabinet. We propose a heuristic, such that optimizing for robot motion based on this heuristic communicates the objective as well as the dynamics constraint to humans. This approach automatically generates expressive attempt motions, and generalizes across different possible failures (Fig. 1.2c). This work is related to prior work on robots learning both the objective function and dynamics from human demonstrations [14]; in our case, we attempt to model how *humans* would infer this from robot behavior.

- Chapter 5 tackles the question of how to give users an understanding of which

[2]Chapter 4 is based on work published in HRI 2018 [13].

states a robot acts correctly in (Fig. 1.2b).[3] In many domains (including driving), the definition of which actions are "correct" is broad for most states. For instance, if an autonomous car is driving without any cars nearby, it does not matter if it slows down or speeds up a little, or turns a little to the right or left—none of these actions will lead to a collision. However, there is a relatively small set of states in which it is extremely important which action the robot takes; we call these *critical states*. In such domains, human end-users just need to know how the robot acts (i.e., its policy) in critical states, and we can communicate this by showing informative examples of robot behavior that cover the set of states that the robot considers to be critical. Whereas communicating objectives only applies to policies that perfectly optimize an objective function, because their tradeoffs must be consistent across situations, this critical-states approach can be applied to a broader range of robot policies, including those that are black boxes.

Part II investigates how transparency can be useful, beyond just increasing comfort and safety in interactions. In Chapter 6, we explore how transparency can improve human teaching of robots.[4]

[3]Chapter 5 is based on work published in IROS 2018 [15].

[4]Chapter 6 is based on work under submission.

Chapter 2

Framework

The main contribution of this thesis is a formalization of what it means to increase transparency of robots to human end-users. In this chapter, we will first lay out the general formalization, and then connect this framework to how we achieve specific aims of transparency in the following chapters.

Let θ^* denote the parameters that fully specify the robot's model—this could be, for example, the weighting on reward features for a robot's objective function, or the parameters of a robot's policy. Then we can denote a person's mental model of the robot as $P(\theta)$: her belief over the possible parameter settings of the robot's model. In this formulation, complete transparency means that the person's mental model of the robot is exactly correct; in other words, that $P(\theta) = \mathbb{1}_{\theta=\theta^*}$.

Our key insight is that a robot's actions change not only the physical environment, but also human observers' mental models (i.e., understanding) of the robot. Motivated by this, we formulate increasing transparency in the framework of a Markov Decision Process (MDP). An MDP is defined by a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ the action space, $\mathcal{P} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ the transition probabilities, $\mathcal{R} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ the reward function, and $\gamma \in (0, 1]$ the discount factor. The following describes each component of our MDP.

- A state $[s, b] \in \mathcal{S}$ consists of both a physical world state s and a belief over θs, which we will use the notation $b(\theta)$ to denote. $\mathcal{S}$ contains all possible physical world states and human beliefs. For example, if the person is considering k possible robot models, then $b \in \Delta_k$, the probability simplex over k dimensions.

- An action $\xi \in \mathcal{A}$ consists of the robot's behavior. This can be either a single action, a, or it can be a sequence of actions $[a_0, \cdots, a_{T-1}]$. In general, actions could be from a variety of modalities, including physical motion, speech, and visualization.

- For the transition probabilities $\mathcal{P}$, we assume we have a model $\mathcal{H}$ that captures how a person will make inference updates to their belief. Recent work in cognitive science proposed a Bayesian framework for human reasoning about the beliefs and objectives of other agents [16–19]. In light of this, we assume that humans perform Bayesian inference over possible robot models, with the expectation that the robot is acting optimally:

$$b'(\theta) \propto b(\theta) P_{\mathcal{H}}(\xi|\theta, s)\,. \tag{2.1}$$

  The choice of $P_{\mathcal{H}}(\xi|\theta)$, the likelihood of action ξ given θ, depends on what θ encodes, as described in the following subsections.

- The reward function $\mathcal{R}$ captures the improvement in the person's mental model of the robot, with respect to the true model, which has parameters θ^*. We define this as the decrease in KL-divergence between the two models:

$$\begin{aligned}\mathcal{R}(b, \xi, b') &= -[D_{\mathrm{KL}}(\mathbb{1}_{\theta=\theta^*} \| b'(\theta)) - D_{\mathrm{KL}}(\mathbb{1}_{\theta=\theta^*} \| b(\theta))] &(2.2)\\ &= \log b'(\theta^*) - \log b(\theta^*)\,. &(2.3)\end{aligned}$$

- For simplicity, we set the discount factor γ to be 1. This means that all rewards (i.e., increases in transparency) are equally valued, whether they occur at the start or the end of a robot trajectory.

We consider a finite-horizon MDP with n steps. In this MDP, the value function of any policy π, starting at state b_1, is

$$V^{\pi}(b_1) = \mathbb{E}[\sum_{t=1}^{n} \gamma^t R(b_t, \xi_t, b_{t+1}) | \pi, b_1]\,. \tag{2.4}$$

Based on our definitions above, the optimal value function (i.e., the best the robot can do in terms of increasing transparency) is thus

$$V^*(b_1) = \max_{\xi_{1:n}} \log b_{n+1}(\theta^*) - \log b_1(\theta^*)\,,^1 \tag{2.5}$$

where b_{n+1} is the human's belief, after starting with a prior belief of b_1 and observing n instances of robot behavior, $\xi_{1:n}$.

[1] This is because after replacing $R(b_t, \xi_t, b_{t+1})$ with $\log b_{t+1}(\theta^*) - \log b_t(\theta^*)$ in the summation of (2.4), the intermediate b_t terms (for $1 < t < n+1$) cancel out in the summation.

Since b_1 does not depend on the robot's behavior, we can optimize for transparency (i.e., obtain maximum reward in this MDP) by choosing examples that satisfy:

$$\arg\max_{\xi_{1:n}} P_{\mathcal{H}}(\theta^* \mid \xi_{1:n}) \,. \tag{2.6}$$

To make more explicit the dependence on $\xi_{1:n}$ and assumption of a model $\mathcal{H}$ of human inference updates, we now switch to the notation $P_{\mathcal{H}}(\theta^*|\xi_{1:n})$ in place of $b_{n+1}(\theta^*)$.

This framework, of leveraging a model of human inference to optimize for examples that maximize their understanding, is closely related to the notion of *legibility* [20] in robotics. Legible motion makes the intentions of the robot (i.e., which goal it is moving toward) clear as early as possible in a trajectory. As in legibility, we focus on robot motion as the mode of communication, because humans naturally and inevitably draw conclusions about a robot based on its motions. We generalize the notion of legibility to optimize for human understanding of a robot's objective function (Chapter 3), its policy (Chapter 5), and its dynamic constraints (Chapter 4), rather than only its goals. In the remainder of this chapter, we will describe how each of these approaches fit into this framework—in particular, what the parameters θ^* refer to, what form the examples ξ take, and how the human's updated mental model $P_{\mathcal{H}}(\theta^* \mid \xi_{1:n})$ is computed.

The form the examples ξ take depends on whether the robot has control over the environment it is placed in. When the environment setup is fixed, then the robot can communicate via exaggerating its motion—humans are good at drawing inferences from motion, as observed by animators [21] and leveraged by prior work in human-robot interaction [20,22,23]. This is the assumption we make when generating expressive failure motions (Chapter 4).

However, if the robot does have control over the setup of the environment (which is often the case in simulation, but rarer in the real world), then it can maximize transparency by both selecting environments based on informativeness and exaggerating its motion within those environments. This is the assumption we make for communicating a robot's objective function (Chapter 3) and its policy (Chapter 5). In these works, we make an additional simplifying assumption, that the robot's actions are optimal rather than exaggerated.

In the following subsections, we describe how to adapt our proposed general framework to achieve these specific aims of transparency. This is part of the contribution of this thesis—we show that these varied goals of transparency in robotics can be tackled via a common approach.

2.1 Communicating Objectives

In Chapter 3, we focus on robots that optimize an objective function that is a linear combination of reward features, with weights θ^* (Sec. 3.3.1). So, θ^* encodes the tradeoffs that a robot makes, and we want to communicate the robot's objectives to human end-users by giving them an (intuitive) understanding of what θ^* is.

Since people naturally get a sense of these tradeoffs as they observe the robot (optimally) acting in a variety of situations, we select examples ξ_E that are this robot's optimal behavior in a particular environment E. By modeling humans as performing Bayesian inference over the space of possible objective parameters, it follows that

$$P_{\mathcal{H}}(\theta^* \,|\, \xi_{E_1:E_n}) = \frac{1}{Z} P(\theta^*) \prod_{i=1}^{n} P_{\mathcal{H}}(\xi_{E_i} \,|\, \theta^*)\,, \tag{2.7}$$

where Z denotes the normalization constant, $\sum_{\theta \in \Theta} P(\theta) \prod_{i=1}^{n} P_{\mathcal{H}}(\xi_{E_i} \,|\, \theta)$. $P_{\mathcal{H}}(\xi_{E_i} \,|\, \theta)$ denotes the likelihood the person assigns to a particular trajectory ξ_{E_i}, if they believe the robot's objective function is parameterized by θ.

Proof-of-Concept Example

As a proof-of-concept, we will apply this to a point robot in a simple grid world environment (Fig. 2.1), where $\theta = [r_1, r_2, r_3]^\top$. Blue squares are terminal states; r_1 and r_2 are the rewards for reaching dark-blue and light-blue squares, respectively. All white squares have a reward of 0 and grey squares have a reward of $r_3 \leq 0$. There are five actions per grid square location (no-op, up, down, left, right) and transitions are deterministic. Taking an action into a wall or after running out of power results in no movement.

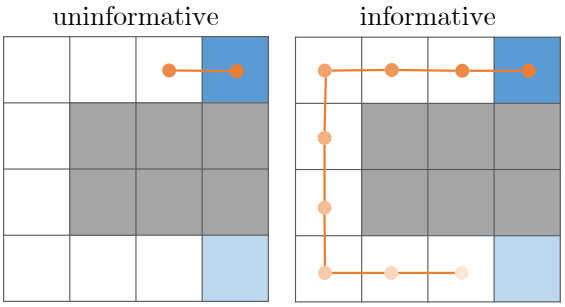


Figure 2.1: The robot's objective is to reach the dark blue square while avoiding grey squares. Assuming the robot acts optimally and can start from any state, starting from the bottom-right in this grid world best communicates its objective.

We choose θ^* to be $[10, 1, -10]$ and set Θ to be all combinations of $r_1 \in \{1, 10\}$, $r_2 = 11 - r_1$, and $r_3 \in \{0, -10\}$. Recall that the robot only demonstrates optimal behavior, so the trajectory ξ_E is the optimal trajectory in environment E with respect to the true robot objective, θ^*. We model $P_{\mathcal{H}}(\xi_E|\theta)$, the likelihood that a person assigns to ξ_E given that the robot

is making tradeoffs θ, as proportional to the exponentiated cumulative discounted reward of the trajectory ξ_E in terms of the objective θ, with a discount factor of 0.9. This is an *exact-inference* model, since we assume the person perceives the exact trajectory ξ_E, can exactly compute the reward of ξ_E with respect to the objective θ, and makes complete inference updates based on this likelihood.

For simplicity, the "environments" that we choose from are each of the possible starting states of the robot in a single grid world, without also varying the grid world itself. Given this restriction, the robot best communicates its objective θ^* by starting from the white square closest to the light blue square. Its optimal behavior is to travel all the way around the grey squares in order to reach the dark blue square—intuitively, we can see that this communicates that the dark blue goal is significantly preferable to the light blue goal, and the grey squares need to be avoided (Fig. 2.1). In contrast, if the robot were to start on the white square next to the dark blue square, then the optimal trajectory would not communicate that grey squares need to be avoided (Fig. 2.2).

Beyond Grid Worlds

Chapter 3 extends this to the domain of autonomous driving, and a larger set of possible environments—with a variety of situations, rather than only altering the starting state in a fixed situation. We also can no longer assume an exact-inference model for the likelihood $P_{\mathcal{H}}(\xi_E|\theta)$, because realistically, human observers make noisy inference updates and cannot perfectly observe the states and actions in a robot trajectory ξ_E. Thus, a key contribution of this work is introducing and evaluating candidate models of how humans compute this likelihood, that incorporate noise from human inference and perception (Sec. 3.3.2). For evaluation, we compute what the optimal teaching examples for each candidate model of human inference are, and then show these examples to real human users and test their resulting understanding of θ^*.

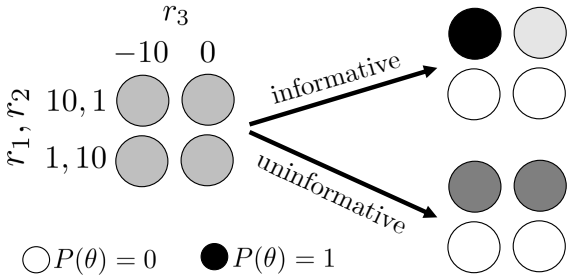


Figure 2.2: After seeing the informative trajectory (Fig. 2.1), the belief is concentrated on the true θ^* ($= [10, 1, -10]$), whereas the uninformative trajectory leaves uncertainty over θs.

2.2 Communicating Dynamic Constraints

In Chapter 4, we focus on robots that fail to complete a task because of dynamics constraints, and we would like to communicate to a human end-user both *what* the robot was trying to do (i.e., its objective) and *why* it is unable to accomplish it (i.e., the dynamics constraints).

Analogously to our approach for communicating objectives, we model humans as performing Bayesian inference over the space of θs, where each θ specifies both the robot's objective and the dynamics constraints it is subject to. Given a trajectory $\xi = [s_0, a_0, s_1, \ldots, a_{T-1}, s_T]$, its likelihood $P(\xi|\theta)$ depends on both the quality and feasibility of the transitions in ξ, with respect to θ:

$$P(\xi \,|\, \theta) = \prod_{i=0}^{T-1} P_{\mathcal{H}}(a_i \,|\, s_i, \theta) P_{\mathcal{H}}(s_{i+1} \,|\, s_i, a_i, \theta)\,. \tag{2.8}$$

The term $P_{\mathcal{H}}(a|s,\theta)$ captures the quality of a transition; a natural choice is for it to be proportional to the exponentiated Q-value[2] of action a in state s, under the objective specified by θ. The term $P_{\mathcal{H}}(s'|s,a,\theta)$ depends on the transition probabilities of the dynamics defined by θ. If the transition (s,a,s') is impossible under these dynamics, then this is zero.

Analogous to (2.7), but with the newly-defined likelihood term that takes into account both the objective and dynamics constraints, humans perform the following inference update:

$$P_{\mathcal{H}}(\theta^* \,|\, \xi) = \frac{1}{Z} P(\theta^*) P(\xi \,|\, \theta^*)\,, \tag{2.9}$$

where Z is the normalization constant, $\sum_{\theta \in \Theta} P(\theta) P(\xi|\theta)$.

Typically, humans can only observe the states in a robot's trajectory, not the actions that the robot takes. For instance, if a robot stops moving after grasping an object placed on a table, it is not clear to a human observer whether the robot is pausing, or whether it is exerting a force on the object that is not large enough to actually move it. Thus, in this case, it is more reasonable to consider a trajectory ξ to be a sequence of states, $[s_0, s_1, \ldots, s_T]$, and we can model humans as marginalizing over the possible actions that could have led from one state to another:

$$P_{\mathcal{H}}(\xi \,|\, \theta^*) = \sum_{a} P_{\mathcal{H}}(a \,|\, s_i, \theta^*) P_{\mathcal{H}}(s_{i+1} \,|\, s_i, a, \theta^*)\, \mathrm{d}a\,. \tag{2.10}$$

[2]The Q-value $Q_\theta(s,a)$ is the expected cumulative discounted reward obtained, if the agent were to take action a in state s and act optimally thereafter with respect to θ.

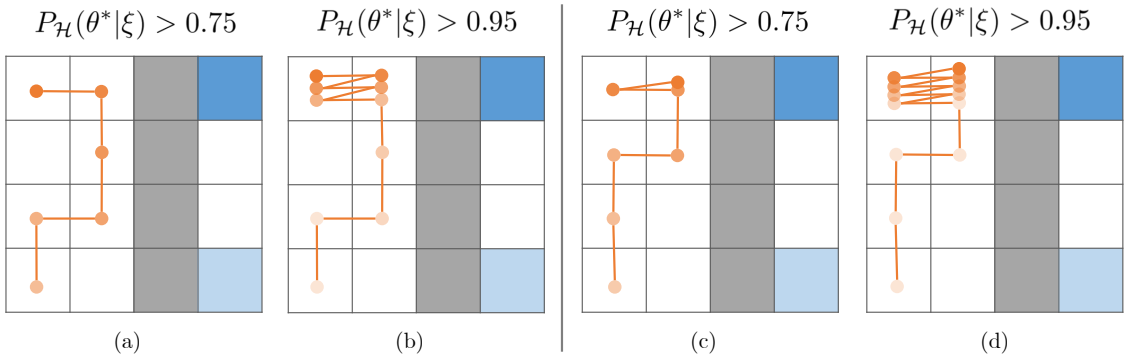


Figure 2.3: Our approach automatically generates an expressive trajectory that communicates θ^*—both the robot's objective function and the dynamics. In this grid world, the robot starts in the lower-left corner, and receives a reward of 10 for reaching the dark-blue square and a smaller reward of 1 for reaching the light-blue square. But the robot cannot cross the dark-grey squares. The posterior belief $P_{\mathcal{H}}(\theta^*|\xi)$ is greater than 0.75 for the orange trajectory ξ shown in (a) and greater than 0.95 for (b), when the true reward is augmented with a proximity reward. The same is true for (c) and (d), but with the true reward and reasoning about pointing.

Proof-of-Concept Example

As a proof-of-concept, we apply this approach to a point robot in a simple grid world environment (Fig. 2.3), where $\theta = [r_1, r_2, \alpha, \beta]^\top$. This is the same as the grid world in Sec. 2.1, except for the dynamics constraints, which are specified by α and β: α is the amount of power the robot has at the start of the episode (i.e., the number of time steps it can act before running out of power), and β is 1 if the dark-grey squares act as walls for the robot and 0 otherwise.

We choose θ^* to be $[10, 1, \infty, 1]^\top$ and set Θ to be all combinations of $r_1 \in \{1, 10\}$, $r_2 = 11 - r_1$, $\alpha \in \{1, 2, \ldots, 9, 10, \infty\}$, and $\beta \in \{0, 1\}$. We use beam search, with a width of $10,000$, to find an expressive trajectory that maximizes $P_{\mathcal{H}}(\theta^*|\xi)$ from Eqn. (2.9). We assume humans compute the likelihood of actions, $P_{\mathcal{H}}(a|s_i, \theta)$, based on a softmax over the corresponding Q-values:

$$P_{\mathcal{H}}(a|s_i, \theta) = \frac{e^{Q_\theta(s_i,a)}}{\sum_{a'} e^{Q_\theta(s_i,a')}}. \tag{2.11}$$

In our grid world, we can compute Q-values exactly via value iteration, with a discount factor of 0.9. But, when the dynamics constraints in θ prevent the robot from reaching all goal locations (as is the case for the chosen θ^*), all Q-values are zero, so there is no incentive for the robot to move at all.

We consider two options for addressing this. The first possibility is that humans believe the robot's proximity to a goal indicates how much the robot values that goal, even if it cannot actually reach the goal. To capture this, we can augment the reward with an additional *proximity reward*,

$$r_{\text{proximity}}(s) = \max_{g \in \mathcal{G}} \left[\left(1 - \frac{\|s - g\|_1}{\max_{s'} \|s' - g\|_1} \right) r(g) \right] , \tag{2.12}$$

where state s and goal g are (x, y) locations in the grid, $\mathcal{G}$ is the set of all goal locations, and $r(g)$ is the reward for goal g. We then compute Q-values based on the augmented reward, $r(p) + \lambda r_{\text{proximity}}(p)$. λ determines the tradeoff between the true reward and the proximity reward; we set this to 0.1 in our experiments. Optimizing for an expressive trajectory that communicates θ^* in this way results in a robot that moves toward the higher-reward goal, in the upper-right corner, and moves back and forth up there—this communicates both which goal the robot prefers, and that it is not able to cross the dark-grey barrier (Fig. 2.3(a) and (b)).

Another possibility is that humans intuitively reason about where a robot's trajectory is "pointing." To capture this, we can define $P_{\mathcal{H}}(a_i|s_i, \theta)$ as the average between the softmax over Q-values (with respect to only the true reward) and the softmax over

$$\sum_{s'} P(s'|s, a, \theta^*) \max \left(0, \frac{(s' - s)^\top (g_{\text{best}} - s)}{\|s' - s\| \|g_{\text{best}} - s)\|} \right) , \tag{2.13}$$

where g_{best} denotes the (x, y) location of the highest-reward goal. Optimizing for an expressive trajectory that communicates θ^* in this way results in similar trajectories as with the first approach (Fig. 2.3(c) and (d)).

Beyond Grid Worlds

In Chapter 4, we apply this approach to generate expressive failure motions on a real robot, that communicate both the robot's objective and the dynamics constraints that prevent it from completing its task. We introduce an approximation based on the proximity reward, taking inspiration from the behavior that emerges in Fig. 2.3(b). We first use trajectory optimization to find an attempt trajectory that is as close as possible to what a successful trajectory would look like, while being subject to the dynamic constraints. Then we augment this with the back-and-forth motion that we see in the grid-world trajectories.

2.3 Communicating Policies

In Chapter 5, instead of assuming that humans model the robot as doing optimization and use that to infer how the robot acts, here we focus on modeling the policy itself: θ^* denotes the parameters of the function approximation for a robot's policy, π_{θ^*}. For a neural network policy, for example, θ^* would be the concatenated network weights of the trained policy. As before, we model humans as performing Bayesian inference over the space of possible θs. For a set of state-action pairs $\{(s_1, a_1), (s_2, a_2), \ldots, (s_n, a_n)\}$, where each $a_i = \pi_{\theta^*}(s_i)$ is the action chosen by the robot's policy, we model humans as inferring

$$P_{\mathcal{H}}(\theta^* | \{(s_i, a_i)\}_{1:n}) = \frac{1}{Z} P(\theta^*) \prod_{i=1}^{n} P_{\mathcal{H}}(a_i \,|\, s_i, \theta^*)\,, \tag{2.14}$$

where Z is the normalization constant, $\sum_{\theta \in \Theta} P(\theta) \prod_{i=1}^{n} P_{\mathcal{H}}(a_i \,|\, s_i, \theta)$. Note that unlike before, we do not constrain these state-action pairs to be from a single trajectory.

Unfortunately, there are no true informative examples that distinguish θ^* from other possible parameters, for expressive policy function approximations. This is because if the policy class has high enough capacity, then each example $(s, \pi_{\theta^*}(s))$ only provides enough information about θ^* to reconstruct how π_{θ^*} acts at state s, leaving the policy's behavior across the rest of the state space undefined. To communicate its θ^*, the robot would need to show how it acts at every single state, which is impossible for large or continuous state spaces.

However, if we look at stochastic policies, our observation is that most tasks do not require taking a specific action, but allow for flexibility in which action to take. Only a few of the states are *critical states*, in which it is very important to take one or a few actions over others. So, we only need to show enough examples to communicate what the critical states of π_{θ^*} are and what actions this policy prefers in those states, and the human can infer that the policy has no strong preference over actions for the other states. In other words, our approach is to show humans which states the robot policy considers to be critical, and inform them that the robot only considers these states and similar ones to be critical. This implicitly communicates that in all the other states, the robot does not consider any action to be important—so this concisely communicates how it would act (i.e., randomly) in the vast majority of possible states.

Proof-of-Concept Example

As a proof-of-concept, consider a point robot in a grid world that is the same as the one in Sec. 2.1, except the robot cannot cross gray squares, and can only move up and to the right. This constraint on movement is analogous to that for a car, which (in most situations) drives in a limited range of forward directions. For simplification, assume that we have trained a policy such that it acts exactly according to a softmax of the Q-values corresponding to a reward of 10 for the dark blue square, 1 for light blue, 0 for white, and a discount factor of 0.9. Then there is a single critical state, that has minimum entropy over actions—the one at which if the policy takes the suboptimal action, there is a "point of no return" in terms of being forced to go to the lower-reward goal (Fig. 2.4).

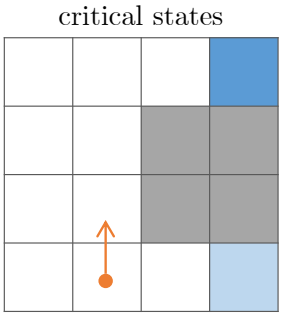


Figure 2.4: The robot can only move up and right, and it cannot cross grey squares. Assuming its policy is trained with a higher reward for the dark blue than light blue goal, then it will have a single critical state (the orange point), at which moving up is significantly preferred to moving right.

Beyond Grid Worlds

In Chapter 5, we apply this approach to communicate neural neural network policies that are trained with maximum-entropy-based reinforcement learning.[3] We further assume that humans generalize to similar states in a way that is consistent with how the robot does—for a neural network policy, this amounts to generalizing based on the distance between internal representations of states (e.g., the output of the last hidden layer). We ensure coverage over the set of the robot's critical states by first clustering states based on this internal representation, and then showing how the robot acts for the most critical state in each cluster.

2.4 Transparency During Teaching

So far, we have focused on robots "teaching," or communicating to, humans something about the robot itself to increase transparency. Typically, the teaching happens the other way around—humans end-users want to teach robots how to do a

[3]Training with a maximum-entropy-based algorithm encourages policies to maximize performance while acting as randomly as possible, so policies are more likely to learn the true critical states of a task, rather than converging to a nearly-deterministic policy.

task according to their preferences. Transparency can also help this teaching process go more smoothly, by either revealing what the robot's current understanding is or revealing what prior (over preferences) the robot started out with, or which feature space it is aware of.

The core assumption of algorithmic teaching is that the teacher has an accurate model of how the learner learns. When this assumption is violated (e.g., when humans are teaching robot learners and do not know how they learn), then transparency on the part of the learner has the opportunity to help the teacher select better teaching examples. We study this both with simulated teachers and real-world human teachers, and find that a simple kind of transparency—the robot indicating what it thinks the human's next action will be—does help teachers identify what the robot does and does not know, and thus select better teaching examples, that lead to faster robot learning.

Part I

Improving Human Mental Models of Robots

Chapter 3

Communicating Robot Objectives

In this chapter, our overarching goal is to efficiently enable end-users to correctly anticipate a robot's behavior in novel situations. When a robot's behavior is a direct result of its underlying objective function, our insight is that end-users need to have an accurate mental model of this objective function in order to understand and predict what the robot will do.

While people naturally develop such a mental model over time through observing the robot act, this familiarization process may be lengthy. Our approach reduces this time by having the robot model how people infer objectives from observed behavior, in order to then show those behaviors that are maximally informative.

We introduce two factors to define candidate models of human inference, and show that certain models indeed produce example robot behaviors that better enable users to anticipate what it will do in novel situations. Our results also reveal that choosing the appropriate model is key, and suggest that our candidate models do not fully capture how humans extrapolate from examples of robot behavior. We leverage these findings to propose a stronger model of human learning in this setting, and conclude by analyzing the impact of different ways in which the assumed model of human learning may be incorrect.[1]

3.1 Motivation and Background

Imagine riding in a self-driving car that needs to quickly change lanes to make a right turn. The car suddenly brakes in order to merge safely behind another car, because it deems it unsafe to speed up and merge in front of the other car.

[1]This work was published as *Enabling robots to communicate their objectives* in RSS 2017 [10] and as an invited submission in AuRo 2019.

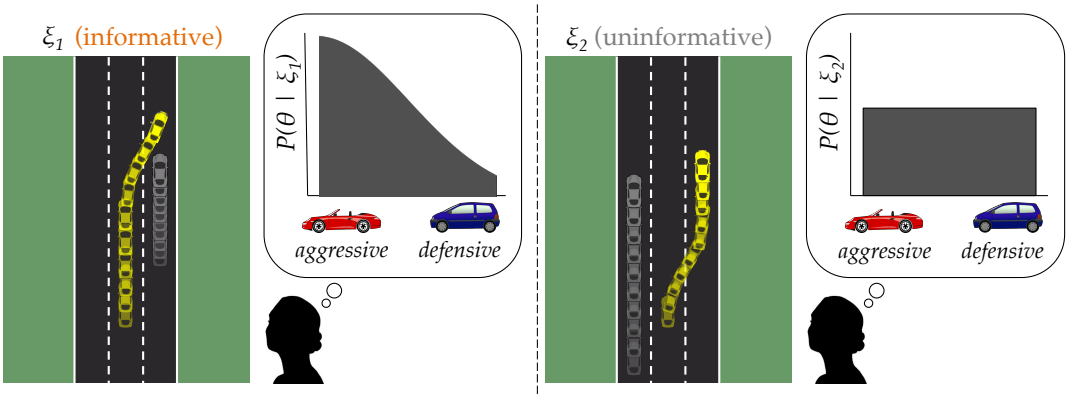


Figure 3.1: We show examples ξ of the yellow autonomous car's behavior that are maximally informative in guiding the human toward understanding the robot's objective function (e.g., aggressive versus defensive). For instance, in environments where the car needs to merge into the right lane, its behavior is more informative when there is another car present (left) than when the lane is empty (right). We consider the case where the robot's objective function is represented by a linear combination of features, weighted by θ.

A passenger who knows this self-driving car is defensive and that it values safety much more than efficiency would be able to anticipate this behavior. But passengers less familiar with the car would not anticipate this sudden braking, so they may be surprised and possibly frightened.

There are many reasons why it is beneficial for humans to be able to anticipate a robot's movements, from subjective comfort to ease of coordination when working with and around the robot [24, 25]. Our goal is to enable end-users to accurately anticipate how a robot will act, even in *novel situations that they have not seen the robot act in before*—like a new traffic scenario, or a new placement of objects on a table that the robot needs to clear.

A robot's behavior in any situation is a direct consequence of the objective (or reward) function the robot is optimizing: (most) robots are rational agents, acting to maximize expected cumulative reward [26]. Whether the robot's objective function is hard-coded or learned, it captures the trade-offs the robot makes between features relevant to the task. For instance, a car might trade off between features related to collision avoidance and efficiency [27], with more "aggressive" cars prioritizing efficiency at the detriment of, say, distance to obstacles [28].

The insight underlying our approach is the following:

The key to end-users being able to anticipate what a robot will do in novel

situations is having a good understanding of the robot's objective function.

Note that understanding the objective function does not mean users must be able to explicate it—to write down the equation, or even to assign the correct reward to a behavior or a state-action pair. Rather, users only need to have an implicit representation of what drives the robot's behavior, i.e., a qualitative understanding of the trade-offs the robot makes.

Fortunately, users will naturally improve their mental model of how a robot acts, given examples of the robot behaving optimally [9]. Further, evidence suggests that people will use this behavior to make inferences about the robot's underlying objective function [17–19], which will enable them to anticipate its behavior in *novel* situations.

A fundamental challenge with this prior work is that it is *passive*: people get exposed to robot behavior in different environments as the robot encounters them. The difference in this work is that we explicitly account for the fact that not all environments are equally informative. In many environments, a robot's optimal behavior does not fully describe the trade-offs that the robot would make in other environments, i.e., parts of the robot's objective will remain under-determined. For example, an autonomous car driving down a highway with no cars nearby will drive at the speed limit and stay in its lane, regardless of its trade-off between efficiency versus staying far away from other cars. Another example is when an autonomous car changes lanes without interacting with any other cars (Fig. 3.1, right). An end-user mainly exposed to these types of behavior will have difficulty forming an accurate mental model of the robot's objective function and anticipating how the robot will behave in more complex scenarios. On the other hand, suppose an autonomous car speeds up to merge in front of another car, cutting it off (Fig. 3.1, left). This scenario more clearly illustrates the trade-offs this car makes regarding safety versus efficiency.

We focus on enabling robots to purposefully choose such *informative* behaviors that actively communicate the robot's objective function. As mentioned in Chapter 1, we envision a training phase for interaction, where the robot showcases informative behavior in order to quickly *teach* the end-user what it is optimizing for. This training phase may need to happen in a simulator, when it is impractical to construct these informative environments in the real world.

In order to choose the most informative example behaviors for communicating a robot's objective function to humans, we take an *algorithmic teaching* approach [29–34]: we model how humans make inferences about the robot's objective function from examples of its optimal behavior, and use this model to generate examples that increase the probability of humans inferring the correct objective function.

The reverse problem, machines inferring objective functions from observed human behavior, can be solved using inverse reinforcement learning (IRL) [12]. Prior work has investigated how to teach an objective function through example behavior to *machine* learners running IRL [35]. But the challenge in teaching *people* instead of machines is that while machines can perform *exact* inference, people are likely to be approximate in their inference. People do not have direct access to configuration-space trajectories and the exact environment state, whereas robots do, at least in kinesthetic teaching (and in [35]). People also cannot necessarily distinguish between a perfectly optimal trajectory for one objective and an ever-so-slightly suboptimal one [36].

In this chapter, our main contribution is to introduce a systematic collection of approximate-inference models and, in a user study, compare their performance relative to the exact inference model. We focus on the autonomous driving domain, where a car chooses example behaviors that are informative about the trade-offs it makes in its objective function. We measure teaching performance—how useful the generated examples are in enabling users to anticipate the car's behavior in test situations—and find that one particular approximate-inference model significantly outperforms exact inference (while the others perform on par). This supports our central hypothesis that accounting for approximations in user inference is indeed helpful, but suggests that we need to be careful about *how* we model this approximate inference.

Further analysis shows teaching performance correlates with covering the full space of strategies that the robot is capable of adopting. For instance, the teaching algorithm cannot just show the car cutting others off; it also needs to show an example where it is optimal to brake and merge behind. We show the best results are obtained by a coverage-augmented algorithm that both leverages an approximate-inference user model and encourages full coverage of all possible driving strategies.

The importance of coverage implies that users are not only learning the robot's objective function and generalizing to novel scenarios based on that; they may also be relying on a memory-based, nearest-neighbor-like approach: directly comparing each novel scenario with the ones they have seen the robot act in, and making inferences based on that. Based on this observation, we study a combined approximate-inference and nearest-neighbor model that better captures how users learn in this domain.

We conclude with a discussion of the implications and limitations of this work. At the forefront of these limitations is our model's assumption of the robot and the human sharing a common understanding of the features that the robot's objective function depends on. We present an analysis of how the results change when this is not true.

This chapter takes a stab at an important yet under-explored problem of commu-

nicating robot objective functions to people. This is important in the short term for human-robot interaction, because it enables transparency and makes robot behavior easier to anticipate. But it is also important in the long term. As AI systems become more capable of optimizing their objective functions, ensuring that these objective functions are aligned with what system designers and end users actually want will become key to ensuring that these systems behave in the intended way [37, 38]. We are hopeful that our work on clarifying these objectives through illustrative examples can help verify objective functions, and contribute toward aligning them with human values. Our results are encouraging, but also leave room for better models of how people extrapolate from observed robot behavior.

3.2 Related Work

A key challenge in our work is modeling how humans make inferences about a robot's objective function from observing its behavior. Humans naturally infer plans, intentions, and goals from the behavior of other agents [39, 40]. Recent work proposed a Bayesian framework for human reasoning, which predicts that humans score candidate hypotheses of an agent's beliefs and desires by the probability that they would have led to the observed behavior, weighted by a prior [16–19]. This probability may depend, for example, on the utility the desire would assign to the observed behavior [19]. This Bayesian framework accurately predicts human reasoning in goal-oriented tasks.

We also model humans as performing Bayesian inference, but we are interested in a trajectory-oriented setting. Even when the goal remains the same (e.g., drive to a particular destination), the trajectory a robot takes depends on the trade-offs it makes in its objective function. Thus, human end-users need to understand not only the robot's goal, but also its objective function. In our work, we introduce approximate-inference models of how humans may score candidate hypotheses for objective functions.

Prior work on enabling humans to better anticipate robot motion has relied on modifying the robot's motions, either by making them more human-like and thus easier to anticipate [41, 42], or by adding anticipatory motion to prepare users for the robot's upcoming action [43, 44]. Our work is complementary to these; it does not require modifying the robot's motion and would improve human anticipation even when the robot moves in a non-human-like way.

Related work explored communicating the payoff matrix in a discrete-action state-invariant game, in which the human observes the reward associated with an action once the robot executes it [45]. What is challenging about our problem

statement is that 1) there are multiple states, and the human needs to generalize the reward to novel state-action pairs, and 2) the human does not get to observe reward directly, but merely sees the robot act, thus all the information they get is that the action was optimal (with respect to discounted cumulative future reward).

Other methods for making robot behavior more transparent include explaining failure modes [46, 47], verbalizing experiences [48, 49], and explaining policies [50].

We leverage algorithmic teaching to choose the most informative examples of optimal robot behavior to show end-users. Prior work on using algorithmic teaching to teach humans primarily focuses on teaching binary classification of images [51–54]. In line with our work, Patil et al. [54] show accounting for human limitations (in their case limiting the number of recalled examples) improves teaching performance.

In our algorithmic teaching framework, we model human learners as running Bayesian IRL [55, 56]: they start with a prior over objective functions that the robot may be optimizing, and update their belief with the likelihood of the observed robot trajectories given the objective function. Prior approaches for Bayesian IRL either rely on exact inference [55] or an action-based likelihood distribution [56]. Action-based distributions have the undesirable effect of favoring trajectories with a smaller action branching factor, whereas trajectory-based distributions do not [57]. Our work considers several trajectory-based distributions, motivated by how humans may perform approximate inference in this domain (Sec. 3.3.4). One of these is equivalent to that in MaxEnt IRL [57].

It is worth noting that many (non-Bayesian) IRL approaches are equivalent to finding the maximum a posteriori estimate in Bayesian IRL, with respect to a particular prior and likelihood function [58]. Modeling learners directly as Bayesian IRL allows us to choose informative example robot behaviors that maximize the posterior probability of the correct objective function while minimizing the probability of others.

3.3 Approach: Algorithmic Teaching of Objectives

We model how people infer a robot's objective function from its behavior, and leverage this model to generate informative examples of behavior.

3.3.1 Preliminaries

Let S be the (continuous) set of states and A be the (continuous) set of actions available to the robot. We assume the robot's objective (or reward) function is

represented as a linear combination of features[2] weighted by some θ^* [59]:

$$R_{\theta^*}(s_t, a_t, s_{t+1}; E) = \theta^{*\top}\phi(s_t, a_t, s_{t+1}; E), \tag{3.1}$$

where s_t is the state at time t, a_t is the action taken at time t, and E is the environment (or world) description. In the case of driving, E contains information about the lanes, the trajectories of other cars, and the starting state of the robot.

A robot's trajectory ξ is defined by a sequence of states s_t and actions a_t for $t = 1, \ldots, T$. Given an environment E, the parameters θ of the objective function determine the robot's (optimal) trajectory ξ_E^θ:

$$\xi_E^\theta = \arg\max_{\xi_E \in \Xi_E} \theta^T \mu(\xi_E), \tag{3.2}$$

where $\mu(\xi_E) = \sum_{t=0}^{T-1} \gamma^t \phi(s_t, a_t, s_{t+1}; E)$ denotes the discounted accumulated feature vector of the trajectory, and γ is a discount factor between 0 and 1 that favors obtaining rewards earlier. Ξ_E refers to all possible trajectories in environment E.

3.3.2 Algorithmic Teaching Framework

We model the human observer as starting with a prior $P(\theta)$ over what θ^* might be, and updating their belief as they observe the robot act. We assume the human knows the features $\phi(\cdot)$ relevant to the task. (This is just our learner model for algorithmic teaching—we put this to the test with real users who do not necessarily have this understanding in Sec. 3.4.3. Further, in Sec. 3.5, we explore what happens when this assumption does not hold.) The robot behaves optimally with respect to the objective induced by θ^*, but as Fig. 3.1 shows, the details of the environment (e.g., locations of nearby cars and the robot's goal) influence its behavior, and therefore influence what effect this behavior has on the person's belief.

To best leverage this effect, we search for a sequence of environments $E_{1:n}$ such that when the person observes the optimal trajectories in those environments, their updated belief places maximum probability on the correct θ, i.e., θ^*:

$$\arg\max_{E_{1:n}} P(\theta^* | \xi_{E_{1:n}}^{\theta^*}) \tag{3.3}$$

To solve this optimization problem, the robot needs to model how examples update $P(\theta^*|\xi_{E_{1:n}}^{\theta^*})$, the person's belief for the robot's true reward parameters. We

[2]We can assume this without loss of generality, as there are no restrictions on how complex these features can be.

propose to model $P(\theta^*|\xi^{\theta^*}_{E_{1:n}})$ via Bayesian inference:

$$P(\theta|\xi^{\theta^*}_{E_{1:n}}) \propto P(\xi^{\theta^*}_{E_{1:n}}|\theta)\, P(\theta) = P(\theta) \prod_{i=1}^{n} P(\xi^{\theta^*}_{E_i}|\theta). \tag{3.4}$$

Computing the normalization factor $\int_{\theta'} P(\xi^{\theta^*}_{E_{1:n}}|\theta')\, P(\theta')$ is intractable, so we approximate this by uniformly sampling candidate θs. Conditional independence can be assumed, since θ contains all the information needed to compute the probability of a trajectory.

With this assumption, modeling how people infer the objective function parameters reduces to modeling the likelihood term $P(\xi|\theta)$: how probable they would find trajectory ξ if they assumed the robot optimizes the objective function induced by θ. We explore different models of this, starting with exact-inference IRL as a special case. We then introduce models that account for the inexactness that is inevitable when real people make this inference.

3.3.3 Exact-Inference IRL as a Special Case

Inverse reinforcement learning (IRL) [12] extracts an objective function from observed behavior by assuming that the observed behavior is optimizing some objective from a set of candidates. When that assumption is correct, IRL finds an objective function that assigns maximum reward (or minimum cost) to the observed behavior.

Algorithmic teaching has been applied to exact-inference IRL learners [35]: the learner eliminates all objective functions which would *not* assign maximum reward to the observed behavior. This can be expressed by the model in (3.4) via a particular distribution for $P(\xi^{\theta^*}_E|\theta)$:

$$P(\xi^{\theta^*}_E|\theta) = \begin{cases} 1, & \text{if } \forall \xi_E, \theta^\top \mu(\xi^{\theta^*}_E) - \theta^\top \mu(\xi_E) \geq 0. \\ 0, & \text{otherwise.} \end{cases} \tag{3.5}$$

This assumes people assign probability 0 to trajectories that are not perfectly optimal with respect to θ, so those candidate θs receive a probability of zero. Thus, each trajectory that the person observes completely eliminates from their belief any objective function that would not have produced exactly this trajectory when optimized. Assuming learners start with a uniform prior over objective functions, the resulting belief is a uniform distribution across the remaining candidate objective functions—θs for which all observed trajectories are optimal.

While this is a natural starting point, it relies on people being able to perfectly evaluate whether a trajectory is *the* (or one of the) *global* optima of any candidate objective function. We relax this requirement in our approximate-inference models.

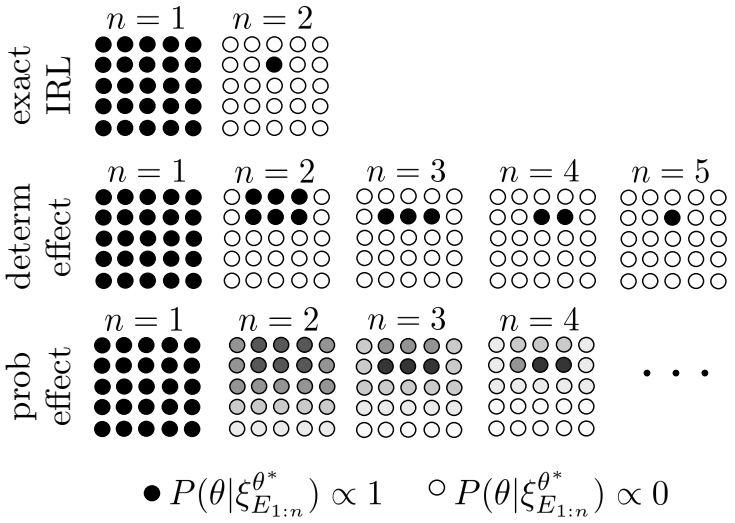


Figure 3.2: A visual comparison of how probabilities of candidate reward parameters θs may be updated for exact-inference and approximate-inference learners, where each dot corresponds to a particular candidate θ. This assumes the models share the same distance metric and the same sequence of examples is shown to each model, until only the true parameters θ^* remain.

3.3.4 Approximate-Inference Models

We introduce a space of approximate-inference models, obtained by manipulating two factors in a 2–by–3 factorial design.

Deterministic versus Probabilistic Effect

In the exact-inference model, a candidate θ is either out or still in: the trajectories observed so far have either shown that θ is impossible (because they were not global optima for the objective induced by that θ), or have left it in the mix, assigning it equal probability as the other remaining θs.

We envision two ways to relax this assumption that a person can identify whether a trajectory is optimal given a θ. One way is for observed trajectories to still either eliminate the θ or keep it in the running, but to be more conservative about which θs get eliminated. That is, even if the observed trajectory is not a global optimum for a θ, the person will not eliminate that θ if the trajectory is *close enough* (under some distance metric) to the global optimum. We call this the *deterministic* effect.

A second way is for observed trajectories to have a *probabilistic* effect on θs: rather than eliminating them completely, trajectories can make a θ less likely, depending

on how far away its optimal trajectory is from the observed trajectory.

In both cases, $P(\xi_E^{\theta^*}|\theta)$ no longer depends on the example trajectory being optimal with respect to θ. Instead, it depends on the *distance* $d(\cdot,\cdot)$ between ξ_E^{θ}, the optimal trajectory for θ, and $\xi_E^{\theta^*}$, the observed trajectory which is optimal given θ^*.

Given some distance metric d along with hyperparameters $\tau, \lambda > 0$,

- For deterministic effect, $P(\xi_E^{\theta^*}|\theta) \propto 1$ if $d(\xi_E^{\theta}, \xi_E^{\theta^*}) \leq \tau$, or 0 otherwise.
- For probabilistic effect, $P(\xi_E^{\theta^*}|\theta) \propto e^{-\lambda \cdot d(\xi_E^{\theta}, \xi_E^{\theta^*})}$.[3]

The deterministic effect results in conservative hypothesis elimination: it models a user who will either completely eliminate a θ or not, but who will not eliminate θs with optimal trajectories close to the observed trajectory. In contrast, the probabilistic effect decreases the probability of θs with far away optimal trajectories, never fully eliminating any (Fig. 3.2).

The exact-inference IRL model (Sec. 3.3.3) is a special case with deterministic effect and a reward-based distance metric with $\tau = 0$; it assumes there is no approximate inference.

Distance Metrics

Both deterministic and probabilistic effects rely on the person's notion of how close the optimal trajectory with respect to a candidate θ is from the observed trajectory. We envision that closeness can be measured either in terms of the reward of the trajectories with respect to θ, or in terms of the trajectories themselves.

We explore three options for d. The first depends on the reward. This distance metric models people with difficulty comparing the cumulative discounted rewards of two trajectories, with respect to a given setting of the reward parameters. So, if in environment E the observed trajectory $\xi_E^{\theta^*}$ has almost the same reward as ξ_E^{θ}, the optimal trajectory with respect to θ, then $P(\xi_E^{\theta^*}|\theta)$ will be high.

- reward-based[4]: $d_r(\xi_E^{\theta}, \xi_E^{\theta^*}) = \theta^\top \mu(\xi_E^{\theta}) - \theta^\top \mu(\xi_E^{\theta^*})$.

The second option depends not on reward, but on the physical trajectories. It assumes it is not high reward that confuses people about whether the observed trajectory is optimal with respect to θ, but rather physical proximity to the true optimal

[3] We noticed normalizing this distribution produced very similar results to leaving it unnormalized, so we do the latter in our experiments, analogous to other algorithmic teaching work not based on reward functions [53].

[4] Note that this is always positive because ξ_E^{θ} has maximal reward with respect to θ.

trajectory: this models people who cannot perfectly distinguish between perceptually-similar trajectories. We measure physical proximity in terms of Euclidean distance, which approximately captures how humans judge perceptual similarity when stimulus dimensions are not easily separable (as is the case for object locations) [60, 61].

- Euclidean-based: $d_e(\xi_E^\theta, \xi_E^{\theta^*}) = \frac{1}{T}\sum_{t=1}^{T} ||s_{E,t}^\theta - s_{E,t}^{\theta^*}||_2$. $s_{E,t}^\theta$ is the state at time t for trajectory ξ_E^θ.

This assumes the two trajectories are the same length, as is the case in our experimental example domain. When trajectories are not the same length, they must first be aligned temporally, for instance via dynamic time warping [62]. In addition, using this distance metric requires an appropriate representation of the state space, e.g., if the dimensions of $s_{E,t}^\theta$ have different ranges, normalization may be necessary. In our example domain, the state is the robot's two-dimensional location.

Finally, a more conservative version of the Euclidean distance metric is the strategy-based metric. The idea here is that for any environment E, trajectories generated by candidate θs can be clustered into types, or strategies. The strategy-based metric assumes people do not distinguish among trajectories that follow the same strategy. For instance, people will consider all trajectories in which the robot speeds up and merges in front of another car to be equivalent, and all trajectories in which the robot merges behind the car to be equivalent. So, if in environment E the observed trajectory and the optimal trajectory with respect to θ have the same strategy, then $P(\xi_E^{\theta^*}|\theta) \propto 1$.

- strategy-based: $d_s(\xi_E^\theta, \xi_E^{\theta^*}) = 0$ if ξ_E^θ and $\xi_E^{\theta^*}$ are in the same trajectory strategy cluster, ∞ otherwise.

Note that the type of effect (deterministic versus probabilistic) does not matter for the strategy-based distance metric, since distances are either 0 or ∞. So, there are a total of five unique approximate-inference models.

Relation to MaxEnt IRL

MaxEnt IRL [57] is an IRL algorithm that assumes demonstrations are noisy (i.e., not necessarily optimal). In our setting, we instead assume demonstrations are optimal but the learner is approximate. These two sources of noise result in the same model: the MaxEnt distribution is equivalent to our *probabilistic reward-based* model:

$$P(\xi_E^{\theta^*}|\theta) \propto e^{\lambda\theta^T\mu(\xi_E^{\theta^*})} \tag{3.6}$$

$$\propto e^{\lambda(\theta^T\mu(\xi_E^{\theta^*}) - \theta^T\mu(\xi_E^\theta))} = e^{-\lambda\cdot d_r(\xi_E^\theta, \xi_E^{\theta^*})}. \tag{3.7}$$

Algorithm 1 Informative example selection

Require: Set of possible environments, $\mathcal{E}$
Require: Robot's reward parameters, θ^*
Require: Set of candidate reward parameters, Θ
Require: Prior over reward parameters, $P_{\mathcal{M}}(\theta)$
Require: Number of examples to select, n
  $\{\hat{P}(\theta)$ keeps track of the learner's belief over reward parameters$\}$
  Initialize $\hat{P}(\theta) \leftarrow P_{\mathcal{M}}(\theta)$
  Initialize $X = []$
  for $i = 1$ **to** n **do**
    for all $E \in \mathcal{E}$ **do**
      $\xi_E^{\theta^*} \leftarrow \arg\max_{\xi_E \in \Xi_E} \theta^{*T} \mu(\xi_E)$
      $Z \leftarrow \sum_\theta P_{\mathcal{M}}(\xi_E^{\theta^*}|\theta)\, \hat{P}(\theta)$
      $P_{\mathcal{M}}(\theta^*|\xi_{X+[E]}^{\theta^*}) = \frac{1}{Z} P_{\mathcal{M}}(\xi_E^{\theta^*}|\theta^*)\, \hat{P}(\theta^*)$
    end for
    $E_i \leftarrow \arg\max_E P_{\mathcal{M}}(\theta^*|\xi_{X+[E]}^{\theta^*})$
    $X \leftarrow X + [E_i]$
    $\hat{P}(\theta) \leftarrow \hat{P}(\theta)\, P_{\mathcal{M}}(\xi_E^{\theta^*}|\theta)$
  end for
  return Sequence of informative examples X

3.3.5 (Submodular) Example Selection

Given a learner model $\mathcal{M}$ that predicts $P_{\mathcal{M}}(\theta^*|\xi_{E_{1:n}}^{\theta^*})$, our approach greedily selects the environment E_t that maximizes $P_{\mathcal{M}}(\theta^*|\xi_{E_{1:t}}^{\theta^*})$. We allow the model to select up to ten examples; it stops early if no additional example improves this probability. Algorithm 1 outlines this approach.

This greedy approach is near-optimal for deterministic effect with a uniform prior, since this makes maximizing $P_{\mathcal{M}}(\theta^*|\xi_{E_{1:t}}^{\theta^*})$ equivalent to maximizing a non-decreasing monotonic submodular function [63]: the total number of candidate θs that are eliminated after observing $\xi_{E_{1:t}}^{\theta^*}$. These two are equivalent because all non-eliminated θs are equally likely (Fig. 3.2). Recall that when deterministic-effect learners observe an example trajectory, they eliminate all candidate θs for which the observed trajectory is not close enough to the optimal trajectory under that θ. The total number of eliminated candidate θs is non-decreasing because adding example trajectories $\xi_E^{\theta^*}$ can only eliminate candidate θs, not add them, and we assume the set of candidate θs considered by the human does not change over time. Additionally, a particular observed trajectory $\xi_E^{\theta^*}$ eliminates the same set of θs no matter when it is

added to the sequence. Thus, showing that example later on in the sequence cannot eliminate more θs than adding it earlier, which makes this function submodular.

3.3.6 Hyperparameter Selection

We would like to select values for hyperparameters τ and λ (for deterministic and probabilistic effect, respectively) that accurately model human learning in this domain. τ and λ affect the informativeness of examples. If τ is too large, then most environments will be uninformative, since the observed trajectory will be within τ distance away from optimal trajectories of many θs, so those θs will not be eliminated. Thus, $P_{\mathcal{M}}(\theta^*|\xi^{\theta^*}_{E_{1:n}})$ will be low no matter which examples $\xi^{\theta^*}_{E_{1:n}}$ are selected. On the other hand, if τ is too small, then some environments will be extremely informative, so only one or a few examples will be selected before no further improvement in $P_{\mathcal{M}}(\theta^*|\xi^{\theta^*}_{E_{1:n}})$ can be achieved. Analogous reasoning holds for λ.

We expect humans to be teachable (i.e., τ cannot be too large) and to have approximate rather than exact inference (i.e., τ cannot be too small), so they would benefit from observing several examples rather than just one or two. Based on this, we select τ and λ for each approximate-inference model by choosing the value in $\{10^{-5}, 10^{-4}, \ldots, 10^4, 10^5\}$ that results in an increase from $P_{\mathcal{M}}(\theta^*|\xi^{\theta^*}_{E_1})$ to $P_{\mathcal{M}}(\theta^*|\xi^{\theta^*}_{E_{1:n}})$ of at least 0.1, and selects the largest number of unique examples to show.

3.4 Experiments

3.4.1 Experimental Domain

We evaluate how our proposed approximate-inference models perform for teaching the driving style of a simulated autonomous car. In this domain, participants witness examples (in simulation) of how the car drives, with the goal of being able to anticipate how it will drive when they ride in it.

Driving Simulator

We model the dynamics of the car with the bicycle vehicle model [64]. Let the state of the car be $\mathbf{x} = [x \;\; y \;\; \theta \;\; v \;\; \alpha]^\top$, where (x, y) are the coordinates of the center of the car's rear axle, θ is the heading of the car, v is its velocity, and α is the steering angle. Let $\mathbf{u} = [u_1 \;\; u_2]^\top$ represent the control input, where u_1 is the change in steering angle and u_2 is the acceleration. Let L be the distance between the front

and rear axles of the car. Then the dynamics model of the vehicle is

$$[\dot{x}\ \ \dot{y}\ \ \dot{\theta}\ \ \dot{v}\ \ \dot{\alpha}] = [v * cos(\theta)\ \ v * sin(\theta)\ \ \frac{v}{L}tan(\alpha)\ \ v * u_1\ \ u_2] \tag{3.8}$$

Environments

We consider a total of 21,216 environments of highway driving configurations (Table 3.1). Each environment has three lanes and a single non-autonomous car. The autonomous car always starts in the middle lane with the same initial velocity, whereas the initial location and velocity of the non-autonomous car varies.

Table 3.1: Parameters of simulated driving environments

Axis of Variation	**Acceptable Values**
Goal	[merge to right, drive forward]
Distance between autonomous and non-autonomous car	[-240, -220,. . . , -100] [100, 120, . . . , 240]
Lane of non-autonomous car	[Left, Center, Right]
Initial velocity, non-autonomous	[20, 25, . . . , 80]
Acceleration time, non-auton.	[0, 0.5, 1, 1.5, 2]
Final velocity, non-auton. car (if acceleration time $\neq$ 0)	[20, 30, 70, 80]

These environments naturally fall into four classes, with two trajectory strategies per class (Fig. 3.3):

- *Merging*: when the non-autonomous car starts in the right lane, and the goal in this environment is to merge into the right lane. The two trajectory strategies are to either speed up and merge ahead of the non-autonomous car, or slow down and merge behind the non-autonomous car.

- *Braking*: when the non-autonomous car starts in the center lane in front of the autonomous car, and the goal is to drive forward. The two trajectory strategies are to either keep driving in the center lane behind the non-autonomous car, or merge into another lane to pass it.

- *Tailgating*: when the non-autonomous car starts in the center lane behind the autonomous car, and the goal is to drive forward. The two trajectory strategies

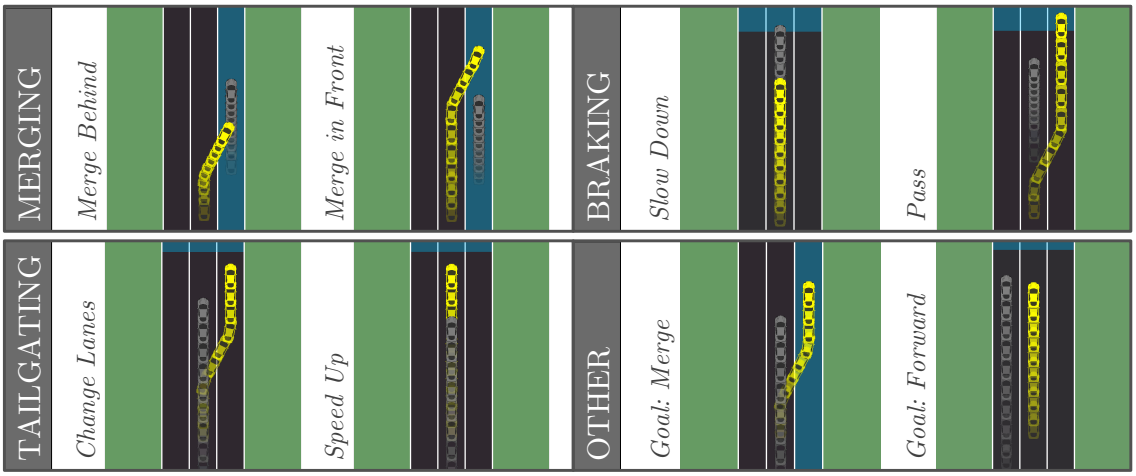


Figure 3.3: The possible driving environments cluster naturally into four classes, with two trajectory strategies per class. Each image shows the trajectories of the autonomous car (yellow) and non-autonomous car (gray) in a particular environment. Positions later in the trajectory are more opaque. The goal of the autonomous car in each environment is highlighted in blue: merge into the right lane or drive forward.

are to either change lanes to avoid the tailgater, or speed up to maintain a safe distance from the tailgater.

- *Other*: all environments not included in one of the first three. The autonomous car is able to reach its goal without any interaction with the other car.

The majority of environments (14,144) are in the *other* class. Of the remaining environments, 3536 are in the *merging* class, 1768 are in *tailgating*, and 1768 are in *braking*.

Reward Features

We use the following reward features $\phi(\cdot)$:

- *distance to other car*: $\sum_{t=0}^{T} \gamma^t \mathcal{N}(p_t | p'_t, \Sigma_t)$, where $p_t = [x_t, y_t]^\top$, p'_t is the position of the non-autonomous car, and Σ_t is chosen such that the major axis is along the non-autonomous car's heading.
- *acceleration*, squared: $\sum_{t=0}^{T-1} \gamma^t \left(v_{t+1} - v_t\right)^2$
- *deviation from initial speed*, squared: $\sum_{t=0}^{T} \gamma^t \left(v_t - v_0\right)^2$

- *turning*: $\sum_{t=0}^{T} \gamma^t \left|\theta_t - \theta_0\right|$
- *distance from goal*: $\sum_{t=0}^{T} \gamma^t \max(0, (x_1 + w) - x_t)^2$ if the goal is to merge into the right lane, and y_T if the goal is to drive forward. w is the width of one lane.

The last four features do not depend on the environment, so we normalize them such that the maximum value of each feature across all trajectories is 1 and the minimum is 0. We use $\gamma = 1$.

Optimal Objective Parameters

We select $\theta^* = [-64 \ \ -0.1 \ \ -1 \ \ -0.1 \ \ -0.5]^\top$, a reward function that is not overly cautious about staying away from other cars. We uniformly sample 1000 candidate θs, and assume learners start with a uniform prior $P(\theta)$.

3.4.2 Analysis with Ideal Users

In Sec. 3.3.4, we proposed five possible approximate-inference user models $\mathcal{M}$. They all model people as judging candidate θs based on the distance between the trajectory they observed and the optimal trajectory with respect to θ, but they differ in what the distance metric is, and whether they completely eliminate candidate θs (deterministic effect) or smoothly re-weight them (probabilistic effect).

Here, we investigate how well algorithmic teaching with these models performs for teaching θ^* to ideal users. First, we generate a sequence of examples for each of our approximate-inference models $\mathcal{M}$, by greedily maximizing $P_{\mathcal{M}}(\theta^* | \xi^{\theta^*}_{E_{1:n}})$, as described in Sec. 3.3.5. We also generate the sequence for the exact-inference model and include a random sequence, for a total of seven sequences.

Run-time

For each of the approximate-inference models, it takes less than a minute to generate this sequence of examples. This includes hyperparameter selection of τ and λ (Sec. 3.3.6). Since we have a fixed set of environments $\mathcal{E}$ and candidate reward parameters Θ, we pre-computed the optimal trajectories and distances between pairs of optimal trajectories $\xi^{\theta^*}_E$ and ξ^{θ}_E for all $E \in \mathcal{E}$ and $\theta \in \Theta$, to speed up example selection.

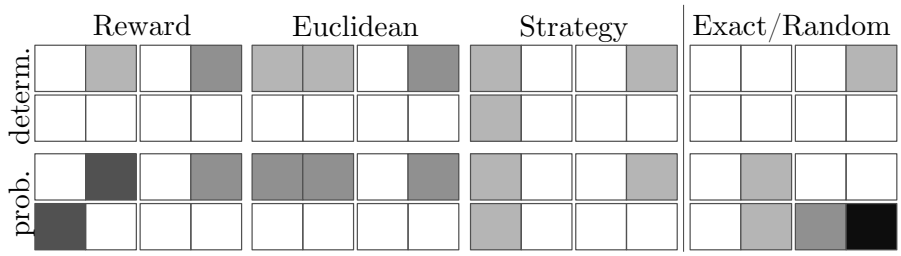


Figure 3.4: The number of examples shown in each of the eight trajectory classes for the approximate-inference models, exact inference model, and random baseline. White = 0 examples shown, and black = 4. The environment classes are arranged in the 2x4 grid as in Fig. 3.3.

Types of Examples Selected

Fig. 3.4 summarizes the types of examples each algorithm selected for its teaching sequence. The exact-inference model selects a single example, because that is enough to completely eliminate all other θs. This works well for an ideal user running exact inference, but our hypothesis is that it does not work as well for real users.

Evaluation with Ideal Users

We evaluate algorithmic teaching on six ideal users, whose learning is precisely exact-inference IRL or one of our five approximate-inference models. We measure, for each "user" $\mathcal{M}$, the probability they assign to the correct objective function parameters, $P_{\mathcal{M}}(\theta^*|\xi^{\theta^*}_{E_{1:n}})$, given $\xi^{\theta^*}_{E_{1:n}}$ from each of the seven generated sequences.

Fig. 3.5 shows the results. First, we see for any ideal user $\mathcal{M}$, the sequence generated by assuming a learner model $\mathcal{M}$ performs best at teaching that user. This is by design—that sequence of examples is optimized to teach $\mathcal{M}$.

Looking across the columns of Fig. 3.5, we see all seven sequences perform equally well for teaching an exact IRL learner (column 1)—even random, because the examples it provides are enough to perfectly eliminate all incorrect θs. This suggests exact IRL does not accurately model real users, whose performance likely varies based on which examples they see. Looking across the rows, we notice assuming a Euclidean distance approximation when generating examples (rows 3 and 5) leads to robust performance across different user models. In the following section, Sec. 3.4.3, we evaluate these generated sequences on real users.

Finally, the random sequence is very uninformative for all ideal users except exact IRL, showing the utility of algorithmic teaching. We explore this utility with real users in Sec. 3.4.4.

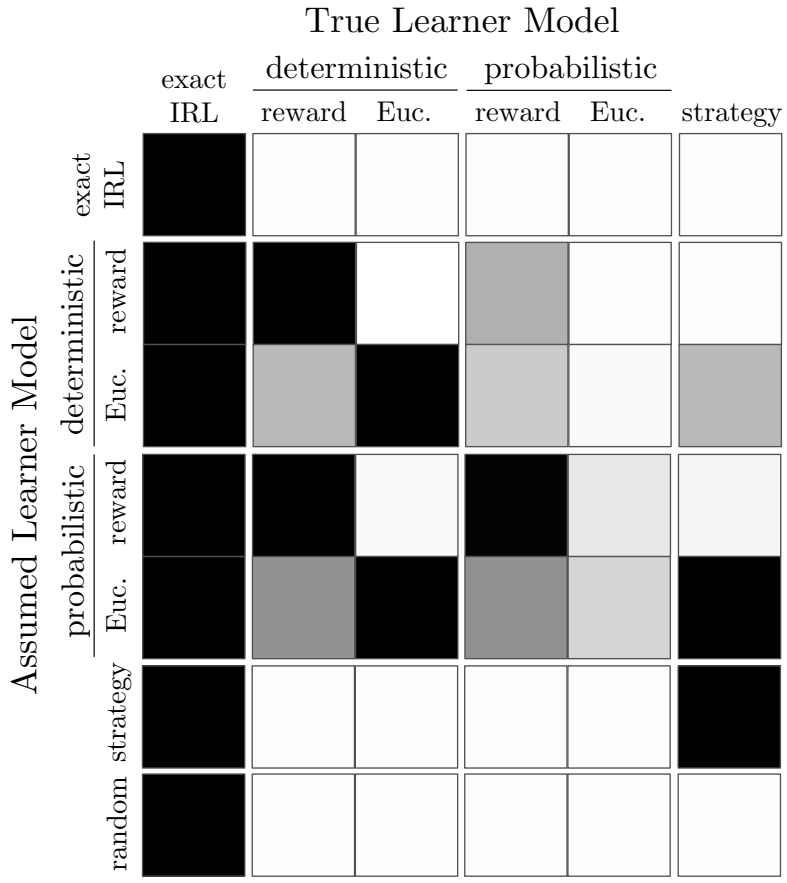


Figure 3.5: The performance of each user model, as evaluated by all other models. White indicates $P_{\mathcal{M}}(\theta^*|\xi^{\theta^*}_{E_{1:n}}) \approx 0$, where $\mathcal{M}$ is the true learner model and environments $E_{1:n}$ are chosen based on the assumed learner model. Black indicates $P_{\mathcal{M}}(\theta^*|\xi^{\theta^*}_{E_{1:n}}) = 1$.

3.4.3 User Study

We now evaluate whether approximate-inference models are useful with real, as opposed to ideal, users.

Experiment Design

Manipulated Variables. We manipulate whether algorithmic teaching assumes exact- or approximate-inference. For the approximate-inference case, we manipulated two variables: the effect of approximate inference (either ***deterministic*** or ***probabilistic***) and the distance metric (***reward-based***, ***Euclidean-based***, or

strategy-based), in a 2–by–3 factorial design, for a total of six approximate inference models. Recall that since the type of effect does not matter for the strategy-based distance metric, there are five unique approximate-inference models.

We show the participant one training environment at a time, in the order that the examples were selected by each algorithm.

Dependent Measures. In the end, we are interested in how well human participants learn a specific setting of reward parameters θ^* from the training examples. Since we cannot ask them to write down a θ, or to drive according to how they think the car will drive (people can drive like themselves, but not so easily like others), we evaluate this by testing each participant's ability to identify the trajectory produced by θ^* in a few test environments. For each test environment, we show the participant four trajectories and ask them to select the one that most closely matches the autonomous car's driving style, and report their confidence (from 1 to 7) for how closely each of the four trajectories matches the driving style.

We have two dependent variables: whether participants correctly identify $\xi_{E_{\text{test}}}^{\theta^*}$ for each test environment E_{test}, and their confidence in selecting that trajectory. We combine the two in a confidence score: the confidence if they are correct, negative of the confidence if they are not—this score captures that if one is incorrect, it is better to be not confident about it.

We use rejection sampling to select test environments in which there are a wider variety of possible robot trajectories. To make sure the four trajectories do not look too similar, we ensure the rewards of alternate trajectories under θ^* are below a certain threshold. In order to not bias the measure, we select one test environment for each of the two trajectory strategy clusters in each of the three informative environment classes, for a total of six test environments. For each test environment, we show two trajectory options in each strategy cluster.

Hypothesis. *Accounting for approximate inference significantly improves performance (as measured by the confidence score).* We leave open which approximate-inference models work well and which do not, since the goal is to identify which captures users' inferences the best.

Subject Allocation. We used a between-subjects design, since examples of the same reward function interfere with each other. We ran this experiment on a total of 191 participants across the six conditions, recruited via Amazon Mechanical Turk. At the end of the experiment, we ask participants what the two possible goals were, to filter out those who were not paying attention. 30 out of 191 (15.7%) answered incorrectly. The average age of the 161 non-filtered participants was 37.0 ($SD = 11.0$). The gender ratio was 0.46 female.

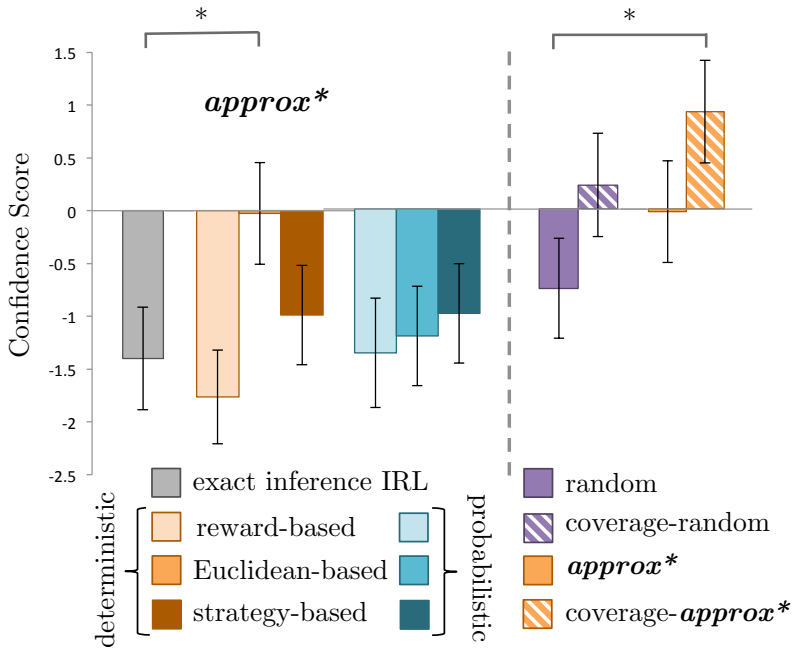


Figure 3.6: Performance of human participants on identifying the autonomous car's trajectory in test environments, after seeing the example trajectories selected by the approximate-inference models (left) and after adding coverage (right) to the sequence of environments selected by the best-performing approximate-inference model, ***approx****. Participants in the coverage-***approx*** condition performed significantly better than those in the random condition. In contrast, enforcing coverage but selecting random sequences (coverage-random) does not lead to statistically significantly better performance.

Analysis

Number of Examples. The number of examples shown depends on which user model is assumed. Exact-inference IRL may produce as few as one example (and does in our case). Approximate-inference models produce more, and random can produce an almost unlimited amount if allowed. Thus, a possible confound in our experiment is the number of examples.

We checked whether this is indeed a confound: do more examples help? Surprisingly, we found no correlation between the number of examples and performance: the sample Pearson correlation coefficient is $r = 0.03$ (Fig. 3.7, left). This suggests that example quantity matters less than example quality.

Approximate-Inference Models. We begin by comparing the approximate-

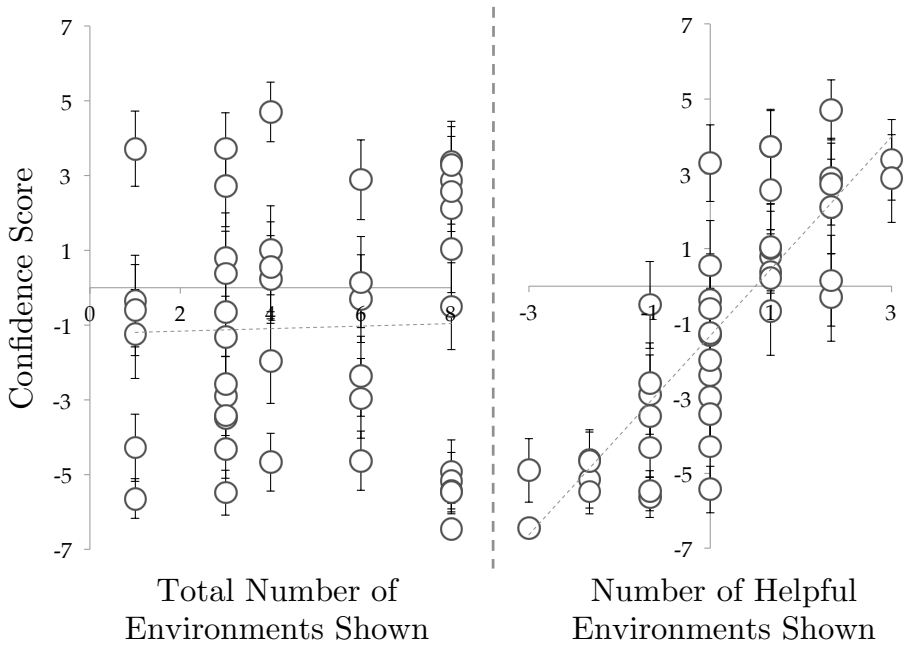


Figure 3.7: **Left:** Lack of correlation between total number of environments shown in a condition, and average participant performance on each test example in that condition. **Right:** Correlation exists between the number of *helpful* training examples shown for an environment class, and average participant performance on the test example corresponding to that environment class.

inference models. We ran a factorial ANOVA on ***confidence score*** with distance measure and determinism as factors (Fig. 3.6, left). We found a marginal effect for ***distance*** ($F(2, 163) = 2.69$, $p = .07$), with Euclidean-based distance performing the best (as suggested by our experiment in Sec. 3.4.2, where Euclidean was the most robust across different ideal users), and reward-based distance performing the worst.

Euclidean-based might be better than reward-based because people decide to keep imperfect θs not when the trajectory they see obtains high reward under that θ, but when it is visually similar to what optimizing for θ would have produced. Euclidean-based might be better than strategy-based because people differentiate between trajectories even when they follow the same strategy. For example, a trajectory that gets very close to another car would be in the same strategy class as a trajectory that stays farther away, as long as they both merge behind the other car, but these two trajectories may give people very different impressions of the car's driving style.

There was no effect for determinism. On average, probabilistic models performed

ever-so-slightly worse than deterministic ones, and the difference was largest for Euclidean distance. This might be because keeping track of what is possible is easier than maintaining an entire probability distribution.

The best-performing approximate-inference model used the Euclidean-based distance with deterministic effects. We refer to this as the ***approx**** model (Fig. 3.6, left).

Hypothesis: Utility of Approximate Inference. We found that despite showing more examples, most approximate-inference models did not perform much better than exact-inference IRL. This shows that not just any approximate-inference model is useful. However, it does not imply that *no* approximate-inference model is useful. To test the utility of accounting for approximate inference, we compared the best model, ***approx****, with exact-inference IRL, and found a significant improvement (Welch's t-test $p = 0.025$).

> *This supports our central hypothesis, that accounting for approximate inference in our model of human inferences about objective functions helps, with the caveat that not just any approximation will work.*

3.4.4 Utility of Algorithmic Teaching

So far, we have tested our central hypothesis, that accounting for approximate IRL inferences can indeed improve the performance of algorithmic teaching of humans in this domain. While this is promising, we also want to test the utility of algorithmic teaching itself: whether our approach is preferable not just to algorithmic teaching with exact inference, but to the robot not actively teaching. Instead, the person must learn from the robot's behavior in environments that it happens to encounter.

Baselining Performance

Baseline Condition. We ran a follow-up study comparing algorithmic teaching with a sequence of optimal trajectories in *random* environments—simulating that the robot does not choose these, but instead happens to encounter them. We recruited 33 users for this condition. The average age of the 28 non-filtered participants was 33.3 ($SD = 9.6$). The gender ratio was 0.46 female.

Controlling for Confounds. There are several variables that could confound this study. First, when generating the random sequence, we might get very lucky or unlucky and generate a particularly informative or uninformative one. To avoid this, we randomly sample 1000 random sequences with the desired number of examples, and sort them based on $P_{\mathcal{M}}(\theta^*|\xi^{\theta^*}_{E_{1:n}})$, where $\mathcal{M}$ is the exact-inference IRL model.

Then we choose the median sequence in that ranking as our random sequence, which will have median informativeness.

Second, different algorithms produce different numbers of examples. For instance, exact-inference IRL only selects one example, which eliminates all θs other than θ^*—because in that environment the optimal trajectories for all θs are at least slightly different than that for θ^*. To give the random baseline the best chance, we choose to select eight environments for it, which is the maximum number of examples shown by any of the other conditions. Since the majority of environments are uninformative (i.e., not in the merging, braking, or tailgating classes), providing the random condition with eight environments is needed to not put it at a serious disadvantage.

Analysis. Algorithmic teaching with our approximate inference model did outperform the random baseline, albeit not significantly (Welch's t-test $p = 0.23$ when comparing participants' confidence scores). Algorithmic teaching *without* accounting for approximate inference actually seems to perform poorly compared to random (Fig. 3.6, right).

Coverage. Digging deeper, we realized users tended to perform well on test cases for strategies in which they had seen a training example. In addition, for each pair of environment strategies A and B (e.g., merge-in-front and merge-behind), if users did not see an example from strategy A but saw one from strategy B, their performance was worse than if they did not see any examples from either A or B! In other words, if users see one trajectory strategy in the training examples and not the opposite strategy, they tend to think the autonomous car will always take the first strategy in that environment type.

Based on this observation, for a particular trajectory strategy A and training environments $E_{1:n}$, we define the number of *helpful (training) environments* for A as:

$$\begin{cases} \sum_{i=1}^{n} \mathbb{1}[h(\xi_{E_i}^{\theta^*}) = A], & \text{if } \sum_{i=1}^{n} \mathbb{1}[h(\xi_{E_i}^{\theta^*}) = A] > 0. \\ -\sum_{i=1}^{n} \mathbb{1}[h(\xi_{E_i}^{\theta^*}) = B], & \text{otherwise.} \end{cases} \tag{3.9}$$

The function h maps a trajectory to the strategy it belongs to, and B is the opposite strategy of A. For instance, if A is the speed-up strategy, then B is the other strategy for the *tailgating* environment, change-lanes. We found a strong correlation between the number of helpful environments shown and users' confidence scores, with a Pearson correlation coefficient of $r = 0.83$ ($p = 1.4 \times 10^{-11}$) (Fig. 3.7, right).

Motivated by this result, we introduce augmented algorithms that ensure coverage of strategies.

Coverage-Augmented Algorithmic Teaching

Since coverage correlates with better user performance, we add a coverage term to our optimization over trajectories $\xi^{\theta^*}_{E_{1:n}}$:

$$\arg\max_{\xi^{\theta^*}_{E_{1:n}}} P_{\mathcal{M}}(\theta^*|\xi^{\theta^*}_{E_{1:n}}) + \nu \sum_c \mathbb{1}[\exists i, h(\xi^{\theta^*}_{E_i}) = c], \tag{3.10}$$

where the sum is over trajectory strategy clusters c.

When greedily selecting the next environment E_t that maximizes Eqn. (3.10), we set

$$\nu = \mathbb{1}[P_{\mathcal{M}}(\theta^*|\xi^{\theta*}_{E_{1:t}}) - P_{\mathcal{M}}(\theta^*|\xi^{\theta*}_{E_{1:t-1}}) < \epsilon],$$

so that only after no examples will significantly increase the probability of θ^*, extra examples are selected to provide coverage across the strategies. We select these extra examples by choosing the best with respect to the approximate-inference model $\mathcal{M}$, to ensure they are informative. Using this approach, we augment our best approximate-inference model, ***approx****, to achieve coverage of all possible strategies.

User Study on Coverage

We next run a study to test the benefit of coverage.

Manipulated Variables. We manipulate two variables: whether we augmented the training examples with coverage, and whether we used a user model to generate the examples or sampled uniformly. We select our best model for the former, ***approx****. From the previous experiment, we have obtained user performance data along the no-coverage dimension—for random and ***approx****—so we run this experiment on only the two new conditions that incorporate coverage: coverage-random and coverage-***approx****. We generate random sequences with coverage by randomly selecting exactly one random environment from each of the eight trajectory strategy classes.

Dependent Measures. We keep the same dependent measures as in our previous user study (Sec. 3.4.3).

Hypothesis. We hypothesize coverage augmentation improves user performance in both conditions, random and ***approx****, compared to the respective conditions without coverage.

Subject Allocation. We used a between-subjects design, with a total of 63 participants across the two conditions. The average age of the 53 non-filtered participants was 34.43 ($SD = 9.0$). The gender ratio was 0.53 female.

Analysis. We ran a factorial ANOVA on ***confidence score*** with coverage and model as factors. We found a marginal effect for coverage ($F(1, 107) = 1.82$, $p = .07$),

suggesting that coverage improves performance. There was no interaction effect, suggesting that coverage helps regardless of using a user model for teaching or not.

Coverage-***approx**** performed best out of the four conditions. The coverage augmentation enabled it to significantly outperform the random baseline (with a Welch's t-test $p = 0.049$), which suggests coverage is useful. Coverage augmentation did not enable the coverage-random condition to outperform the random baseline ($p = 0.159$), which suggests the approximate-inference model is useful (Fig. 3.6, right).

Overall, coverage alone helped, but was not sufficient to outperform the random baseline. From the previous experiment, we know that the approximate-inference user model helped, but was also not sufficient to outperform this baseline. The improvement is largest (and significant, modulo compensating for multiple hypotheses) when we have a coverage-augmented approximate-inference IRL model.

> *When leveraged together, coverage with the right model of approximate inference have a significant teaching advantage over random teaching, as well as over IRL models that assume exact-inference users.*

3.5 Analysis of Alternative Learner Models

Algorithmic teaching relies on having a reasonably accurate model of how the learner learns. For instance, we found that when accounting for approximate inference in the learner model, most approximations did not perform significantly better than the exact-inference IRL baseline, although the best model did (Sec. 3.4.3). In this section, we will reconsider several key assumptions that our approach makes about human learning of objective functions, and analyze how teaching effectiveness is affected when they do not hold. These assumptions are that human end-users (1) pay attention to exactly the same set of features that define the robot's objective function, (2) adhere to a particular setting of hyperparameters for approximate inference, and (3) use only their understanding of the robot's objective function to anticipate its behavior.

3.5.1 Feature Mismatch

Recall that we model human end-users as updating their belief over candidate θs as they observe the robot act (Sec. 3.3.2). Each candidate θ corresponds to a particular setting of trade-offs between the reward features $\phi(\cdot)$ considered by the robot. The set of candidates includes θ^*, the true parameters for the objective function optimized by the robot.

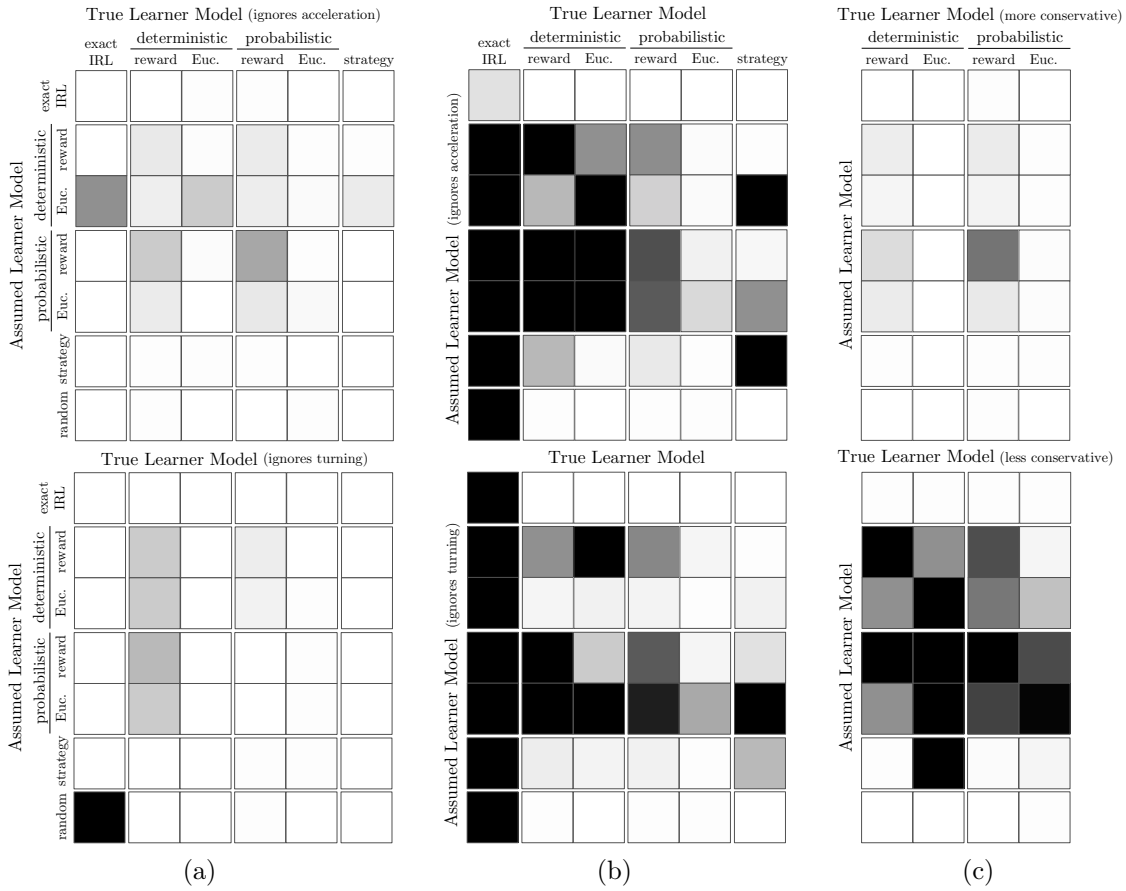


Figure 3.8: The performance of each user model, as evaluated by all other models. The assumed and true learner models differ in terms of either features considered or hyperparameters. White indicates $P_{\mathcal{M}}(\hat{\theta}^*|\xi^{\theta^*}_{E_{1:n}}) \approx 0$, where $\mathcal{M}$ is the true learner model and environments $E_{1:n}$ are chosen based on the assumed learner model. Black indicates $P_{\mathcal{M}}(\hat{\theta}^*|\xi^{\theta^*}_{E_{1:n}}) = 1$. (a) The assumed learner model considers all (five) features $\phi(\cdot)$, but the true learner model ignores one of them. (b) The true learner model considers all features, but the assumed learner model ignores one of them. (c) Both the assumed and true learner models consider all features. However, the true learner model is either more or less conservative in the elimination or downweighting of candidate θs than the corresponding assumed learner model.

But what if end-users do not pay attention to exactly the same reward features $\phi(\cdot)$ as the robot does? In the context of our autonomous driving example domain, perhaps they do not notice sudden braking (the *acceleration* feature) or unnecessary

lane changes (the *turning* feature). To determine how much of an impact feature mismatch has, we analyze how well algorithmic teaching performs for teaching θ^* to ideal end-users, that reason over either a subset or superset of the features that define the robot's objective function.

Note that when users reason over a different set of reward features $\hat{\phi}(\cdot)$, the space of candidate $\hat{\theta}$s no longer includes the true parameters θ^*. So, the best we can do is to teach users the true trade-offs between the reward features in common for the users' and robot's objective functions. Let $\hat{\Theta}^*$ denote the set of all correct candidates, that have the same tradeoffs as θ^* for these shared reward features. As in Sec. 3.4.2, we evaluate teaching performance by measuring, for each "user" $\mathcal{M}$, the probability they assign to the correct objective function parameters ($\max_{\hat{\theta}^* \in \hat{\Theta}^*} P_{\mathcal{M}}(\hat{\theta}^* | \xi^{\theta^*}_{E_{1:n}})$), given $\xi^{\theta^*}_{E_{1:n}}$ from each of the generated sequences of examples.[5] Unlike in Sec. 3.4.2, these ideal users no longer correspond to one of the models used to select examples.

Missing Features

We first greedily maximize $P_{\mathcal{M}}(\theta^* | \xi^{\theta^*}_{E_{1:n}})$ to generate a sequence of teaching examples for each of our five approximate-inference models $\mathcal{M}$ and the exact-inference model, as well as generate a random sequence (same as those in Sec. 3.4.2). We then take our original set of five features (Sec. 3.4.1) and remove either the acceleration or turning feature, resulting in two subsets of features that our ideal users could reason over. We always keep the features for distance to other car, distance from goal, and deviation from initial speed (i.e., the speed limit) since these are fundamental to the driving task. There is a single true reward parameter $\hat{\theta}^*$ that matches the trade-offs of θ^* in the subset of features considered by the user.

We found that when users only pay attention to a subset of features, teaching is much less effective (Fig. 3.8a). This may be because our original set of five features does not contain any redundant ones. So, for many environments, the optimal trajectory for θ^* is different than that for $\hat{\theta}^*$.[6]

Consequently, we see that now almost no sequence of examples is able to teach the exact-inference IRL learner. Since for many environments E, the observed trajectory $\xi^{\theta^*}_E$ is not optimal for $\hat{\theta}^*$, this leads to $P(\xi^{\theta^*}_E | \hat{\theta}^*) = 0$ for exact-inference IRL learners and the true reward parameter $\hat{\theta}^*$ is then eliminated. For similar reasons, it is no

[5] We take the maximum probability assigned to a correct candidate because we would be happy with the user learning any of these.

[6] When $\hat{\theta}^*$ ignores the acceleration feature, the optimal trajectory differs in 50% of the environments selected as teaching examples, and the strategy of the optimal trajectory differs in 21%. When $\hat{\theta}^*$ ignores the turning feature, the optimal trajectory differs in 50% and the strategy differs in 14%. (Two trajectories are the same if they consist of the same sequence of states.)

longer the case that for any ideal user model $\mathcal{M}$, the sequence generated by assuming that learner model performs best at teaching that user.

Out of all the models, it seems deterministic Euclidean is slightly more robust for teaching users who reason over a subset of features. In particular, when (ideal) users expect that the robot does not consider the acceleration feature (e.g., gradually braking and slamming on the brakes are equally fine), the results are similar to those in our user study (Sec. 3.4.3): the sequence of examples generated by assuming deterministic, Euclidean-based approximate-inference is most effective at teaching (Fig. 3.8a, top, row 3). This suggests that the real users in our study may have overlooked acceleration of the autonomous car—this is reasonable, since they were viewing the car's trajectory as a video; acceleration would have been much more apparent if users sat in a realistic driving simulator with force feedback.

Additional Features

To measure the effect of users reasoning over a superset of the features that determine the robot's objective function, we alter the robot's objective function to be a linear combination over four of the original five reward features $\phi(\cdot)$, ignoring either the acceleration or turning feature. For each of these two subsets of features, we recompute a sequence of teaching examples for each of the approximate-inference models $\mathcal{M}$ and the exact-inference model by greedily maximizing $P_{\mathcal{M}}(\theta^*|\xi^{\theta^*}_{E_{1:n}})$, as well as generate a random sequence.

Our ideal users reason over the full set of features. Since the robot's goal is to communicate the tradeoffs between only the features that it is paying attention to, there is no restriction on the reward weights corresponding to features that the robot is *not* paying attention to. So, there are multiple correct reward candidates $\hat{\theta}^*$ considered by the user, that match the trade-offs of θ^* in the subset of features considered by the robot.

We found that when users pay attention to a superset of features, teaching is as effective as when users pay attention to the exact set of features that the robot does (Fig. 3.8b and Fig. 3.5, respectively). This may be because in our problem setup, we are equally happy with the user learning any of the correct candidates in $\hat{\Theta}^*$, which may not be the case in practice.

3.5.2 Approximate-Inference Hyperparameter Mismatch

Recall that our proposed approximate-inference models require setting a hyperparameter: τ for deterministic effect, λ for probabilistic effect. This hyperparameter controls how conservative the learner is in eliminating or downweighting candidate θs

(Sec. 3.3.6). More conservative means that learners eliminate fewer θs or downweight them by less, which corresponds to a larger τ and a smaller λ. Less conservative means the opposite, and corresponds to a smaller τ or a larger λ. We originally chose the hyperparameter for each approximate-inference model $\mathcal{M}$ with a heuristic, to be similar to how humans might learn (Sec. 3.3.6).

We consider teaching ideal users that are either more or less conservative, by a factor of two for τ and λ, in how they eliminate or downweight candidate reward parameters. The teaching sequences are the same as in Sec. 3.4.2. As one might expect, when users are more conservative in eliminating or downweighting θs, they become less teachable: even after observing multiple example trajectories, they do not prune their space of candidate θs by much, so $P(\theta^*|\xi^{\theta^*}_{E_{1:n}})$ is close to zero across all models and teaching sequences (Fig. 3.8c, top). On the other hand, when users are less conservative, they tend to perform well across all teaching sequences computed for approximate-inference models (with a reward- or Euclidean-based distance metric). However, these ideal users are still unable to learn from a random sequence or the single example computed to teach exact-inference IRL learners (Fig. 3.8c, bottom).

3.5.3 Hybrid Models

A key assumption underlying our approach is that human end-users use their understanding of the robot's objective function to anticipate the robot's behavior. This understanding is what enables them to generalize and anticipate the robot's behavior in *novel* scenarios, that are different than those they have seen the robot act in previously (e.g., the test environments in our user studies).

However, the importance of coverage implies that users are unable to generalize perfectly, since they need to see all possible strategies of robot behavior. This suggests something else is going on: users may also be learning the robot's policy directly (analogous to robots using imitation learning to learn from human demonstrations), or they may be learning objective functions that are complex and environment-dependent, and therefore difficult to generalize across test cases. Alternatively, people may use hierarchical reasoning, in which they first determine which trajectory strategy the robot will take (for which they need examples of each possible strategy), and then select the most likely trajectory within that strategy cluster.

Along the lines of direct policy learning, we explore the possibility that users are learning a policy that maps from environment to the trajectory taken by the robot, by relying on a nearest neighbor approach: directly comparing each novel environment with the ones they have seen the robot act in, and making predictions based on that. This is motivated by the observation that humans seem to use such a nearest-neighbor, or *exemplar*-based, approach in other domains, in particular

category learning [65, 66].

We hypothesize that a hybrid model, that combines approximate inference with exemplar-based reasoning, will better capture how humans learn in this domain. We will first introduce a framework for combining these two types of learning, and then evaluate how well a hybrid model correlates with responses from our user studies.

Instead of modeling people as solely performing inference over objective function parameters, we can model them as computing a distribution over possible trajectories for a new environment E_t:

$$P(\xi_{E_t}|\xi^{\theta^*}_{E_{1:n}}).$$

If humans compute this distribution by only reasoning over objective function parameters (as we have assumed up until now), then we have

$$P_{\mathcal{M}}(\xi_{E_t}|\xi^{\theta^*}_{E_{1:n}}) \propto \mathbb{E}_{\theta \sim P_{\mathcal{M}}(\theta|\xi^{\theta^*}_{E_{1:n}})} P(\xi_{E_t}|\theta) \tag{3.11}$$

In contrast, if humans compute this distribution through exemplar-based reasoning, then we have

$$P(\xi_{E_t}|\xi^{\theta^*}_{E_{1:n}}) \propto \sum_{i=1}^{n} g_E(E_t, E_i) g_T(\xi_{E_t}, \xi^{\theta^*}_{E_i}) \tag{3.12}$$

where g_E and g_T are similarity metrics between environments and robot trajectories, respectively.

One way to combine these two models into a hybrid model is to take the smaller of the two values:

$$\begin{aligned} P(\xi_{E_t}|\xi^{\theta^*}_{E_{1:n}}) \propto \ \min \big(& \mathbb{E}_{\theta \sim P(\theta|\xi^{\theta^*}_{E_{1:n}})} P(\xi_{E_t}|\theta), \\ & \sum_{i=1}^{n} g_E(E_t, E_i) g_T(\xi_{E_t}, \xi^{\theta^*}_{E_i}) \big). \end{aligned} \tag{3.13}$$

This models end-users as being conservative: both lines of reasoning must indicate that a particular trajectory is likely, in order for people to anticipate that the robot will take that trajectory.

Evaluation

Recall that in our user studies, we showed participants a sequence of autonomous car trajectories $\xi^{\theta^*}_{E_{1:n}}$ (that varied across conditions) and then asked them to assign a confidence rating for each of four trajectories in a new environment E_t, based on whether they thought it was the same autonomous car driving for that new trajectory.

These confidence ratings capture how likely a participant thinks that trajectory is in environment E_t, given the observed examples of robot behavior. So, we can evaluate models $\mathcal{M}$ of human learning based on how well their predicted values of $P_{\mathcal{M}}(\xi_{E_t}|\xi^{\theta^*}_{E_{1:n}})$ correlate with the confidence ratings from our user studies, averaged across all users in the same condition. The (five) approximate-inference models all have a weak correlation, with Pearson correlation coefficients r between 0.34 and 0.40 (p-values $< 10^{-5}$).

For the exemplar-based model, we set $g_E(E_a, E_b)$ to 1 if E_a and E_b are in the same environment class (e.g., merging), and 0 if they are not. This captures that when faced with a novel environment, people might recall what they observed the robot doing in other environments with the same class, and use that to anticipate what the robot will do in the new environment. We set $g_T(\xi_a, \xi_b)$ to 0 if ξ_a and ξ_b are not in the same trajectory strategy cluster, and to $d_e(\xi_a, \xi_b)$ if they are. This captures that people consider a trajectory more likely if it shares the same strategy and is similar (in terms of Euclidean distance) to one they have observed in the past. In particular, people only realize the robot is capable of executing a particular strategy if they have previously observed the robot doing it. This exemplar-based model also correlates weakly with the user data, with a Pearson correlation coefficient r of 0.40 (p-value $< 10^{-5}$).

This particular exemplar-based model relies on having a reasonable specification of environment classes and trajectory strategies. In our experimental domain, these were defined based on hand-picked rules, but they could be automatically computed instead. Environment classes can be obtained automatically by clustering the environments. Trajectory strategies can be obtained automatically by clustering the trajectories in each environment class, e.g., based on trajectory feature vectors $\mu(\xi_E)$. Or, trajectory strategies could correspond to the homotopy classes in an environment.

Our proposed hybrid model (Eqn. (3.13)) correlates more strongly with the user data than either the approximate-inference or exemplar-based models alone, with Pearson correlation coefficients r between 0.40 and 0.61 (p-values $< 10^{-5}$). This suggests that participants rely on both modes of learning in this domain.

3.6 Discussion

Summary

We take a step toward communicating robot objective functions to people. We found that an approximate-inference model using a deterministic Euclidean-based update on the space of candidate objective function parameters performed best

at teaching real users, and outperforms algorithmic teaching that assumes exact inference. After augmenting such a model with a coverage objective, it outperformed letting the user passively familiarize to the robot.

We additionally provide an empirical analysis of alternative learner models, in which we evaluate how mismatch between the assumed learner model and the true model impacts the effectiveness of algorithmic teaching, and propose a combined approximate-inference and exemplar-based model that better captures how users learn in this domain.

Implications

Making robot objective functions more transparent to people is a worthwhile goal to work towards. As robots become increasingly capable and are deployed in more situations, understanding what a robot is optimizing for (and thus being able to anticipate how it will move around in its environment) is key to safe, comfortable, and coordinated human-robot interaction.

As an example of a concrete future use case, imagine allowing users to tune a knob that controls the objective function a robot optimizes. This knob would likely not correspond to the true reward features; it may instead collapse them onto one or two interpretable axes. For instance, for an autonomous car, this knob may control how efficient versus defensive the car drives. Each point on the dial would correspond to a different setting of reward parameters. Our approach could be used to efficiently communicate the behavior corresponding to each point, so passengers can choose the one they are most comfortable with.

Limitations and Future Directions

Our results reveal the promise of algorithmic *teaching* of robot objective functions. Future work could apply this framework to teach more complex objective functions, for instance those that reason about the intentions and goals of other agents, or how other agents will respond to the robot's behavior (as in [67]).

However, the coverage results suggest that an IRL-only model is not sufficient for capturing how people extrapolate from observed robot behavior. We took a step toward investigating whether users may be directly learning policies in addition to reasoning about the robot's objective function. There is more work to be done on finding accurate models of human learning in this domain.

Our analysis also suggests that it is important for the robot and end-users to achieve common ground on which features are important. This analysis was limited to end-users considering either a subset or superset of the features that define the

robot's objective function. Future work could study more complex differences in features considered, or investigate interactions for achieving common ground on which features to pay attention to.

Furthermore, in this work we focus on the robot's physical behavior as a communication channel because people naturally infer utility functions from it. Future work could augment this with other channels, such as visualizations of the objective function or language-based explanations.

Finally, this work applies only to robots that behave optimally with respect to their underlying objective function. So, our approach cannot be applied directly to robot policies trained with imitation learning or reinforcement learning, since they are not guaranteed to behave optimally with respect to any objective function. The next chapter explores how it is still possible to show informative examples of robot behavior that give human end-users better mental models of black-box neural network policies, trained with deep reinforcement learning.

Chapter 4

Expressing Robot Incapability

In the previous chapter, we assumed that robots can successfully accomplish the task at hand, but this is not always the case. It is useful for humans to understand *why* a robot has failed, in order to be better equipped to assist the robot, and gain an understanding of other situations that this robot would fail in. Thus, the goal of this chapter is to enable robots to express their incapability, and to do so in a way that communicates both *what* they are trying to accomplish and *why* they are unable to accomplish it. We frame this as a trajectory optimization problem: maximize the similarity between the motion expressing incapability and what would amount to successful task execution, while obeying the physical limits of the robot. We introduce and evaluate candidate similarity measures, and show that one in particular generalizes to a range of tasks, while producing expressive motions that are tailored to each task. Our user study supports that our approach automatically generates motions expressing incapability that communicate both *what* and *why* to end-users, and improve their overall perception of the robot and willingness to collaborate with it in the future.[1]

4.1 Motivation and Background

As robots become increasingly capable, they may unintentionally mislead humans to overestimate their capabilities [68]. Thus, it is important for a robot to communicate when it is incapable of accomplishing a task. There are two relevant pieces of information when expressing incapability: *what* the task is, and *why* the robot is incapable of accomplishing it.

Understanding *why* the robot is incapable gives observers a better understanding

[1]This work was published as *Expressing robot incapability* in HRI 2018 [13].

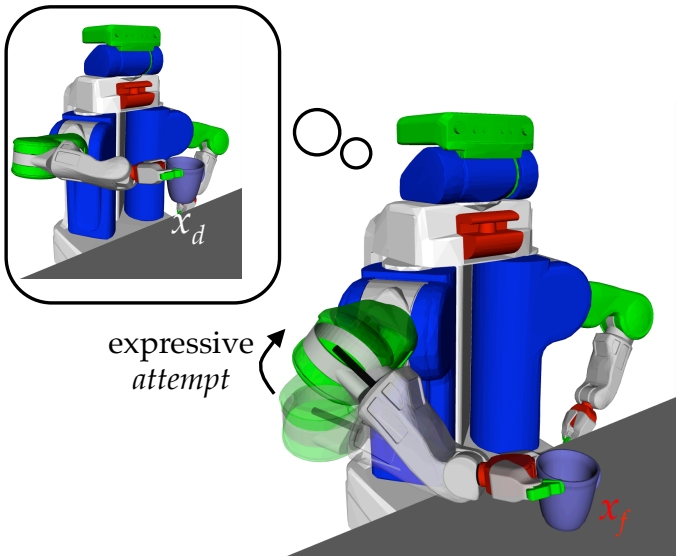


Figure 4.1: We introduce a method to generate motion for incompletable tasks that communicates both the intended goal of the task and why the robot is incapable of completing the task. The method generates an *attempt* motion meant to resemble successful execution (e.g., moving the end-effector from x_f to x_d) while obeying the constraints on the robot's limitations. In this example, the robot ends up lifting its elbow to communicate that it is trying to lift the cup, but the cup is too heavy for it.

of its capabilities, which improves joint human-robot task performance [45]. Transparency about the causes of incapability also helps observers assign blame more accurately [69]. If observers also understand *what* the robot was trying to do, they are better able to help the robot complete the task [47, 70, 71].

One of the simplest ways to express incapability is to carry out the failure. Unfortunately, not all failures are inherently communicative about the *what* and the *why*. The fact that the robot failed to complete the task means that it might not have gotten far enough in the task for the *what* to become obvious—in fact, robots sometimes fail before they even start. Our goal in this work is to expressively show a robot's incapability, beyond simply failing.

In general, even just acknowledging an incapability (e.g., via language or motion) mitigates the damage to human perception of the robot [22, 72, 73]; in some situations, demonstrating an incapability may actually increase the likeability of the robot [74, 75], due to the Pratfall Effect [76]. If we can further ensure better understanding, we hope that people will not only evaluate the robot more favorably, but they will also be able to make more accurate generalizations of the robot's capability across

different tasks.

We focus on the robot's motion as the communication channel, which has already been established as an effective and natural way of communicating a robot's intent [20].

> *Our key insight is that a robot can express both* what *it wants to do and* why *it is incapable of doing it by solving an optimization problem: that of executing a trajectory similar to the trajectory it would have executed had it been capable, subject to constraints capturing the robot's limitations.*

Take lifting a cup that is too heavy as an example, or turning a valve that is stuck. Once the robot realizes that it is incapable of completing the task, the robot would find some motion that still conveys what the task is and sheds light on the cause of incapability, all without actually making any more progress on lifting the cup or turning the valve. We call this motion an *attempt*; Fig. 4.1 shows what an attempt might look like for the lifting example.

We focus on situations like these, in which the robot is unable to accomplish a task due to dynamics constraints, that prevent it from moving its end-effector (and the object it is manipulating) to the desired goal pose. Although this is a relatively narrow set of tasks that the robot may be incapable of completing, it is a step toward *automatically* generating task-specific motions expressing incapability: prior work relied on either simple strategies [70, 77] or motions hand-crafted for each specific situation in which the robot is incapable [22].

Our main contribution is to frame the construction of expressive incapability trajectories as a trajectory optimization problem, motivated by our framework for increasing transparency (Chapter 2). We explore several reasonable objectives for this optimization problem, and find that one generalizes best across a range of incompletable tasks. Our user study shows that attempt trajectories significantly improve not only participants' understanding of what the robot is trying to do and why it cannot, but also their overall perception of the robot and willingness to collaborate with it in the future.

4.2 Related Work

Our motions communicate both the what and the why, i.e., both intent and the cause of incapability.

What/Intent

Much work has focused on motion for conveying robot intent, i.e., what task the robot is doing [20, 43, 44]. What is different in our work is that the robot is incapable of actually doing the intended task, so it needs a way to convey enough about the task without being able to actually do it. This is our idea of *attempt*: in our work, the robot generates a (failed but expressive) attempt at the task. We focus on how to autonomously generate such attempts.

Communicating intent is useful: legible motion improves joint human-robot task performance [25] and in fact arises naturally from optimizing for joint performance [78]. Beyond motion, prior work has explored communicating intent through visualizing planned trajectories (e.g., via targeted lighting [79] or augmented reality [80, 81]), gaze [82], body language [83], human-like gestures [84, 85], verbal communication [86], and LED displays [87].

Why/Cause of Incapability

Prior work on using motion to communicate *why* a robot cannot complete a task relied on simple strategies: moving back-and-forth when stuck in front of an obstacle [77], or repeatedly executing a failing action [70]. In the context of lifting a cup that is too heavy, the latter approach would result in a trajectory that repeatedly reaches for the cup, grasps it, then rewinds. We show in Sec. 4.4.3 that our approach significantly improves identification of the task goal and cause of incapability, compared to the latter method. Another approach relies on hand-designed motions, crafted per-task using animation principles, to indicate recognition of success or failure in completing the task [22]. In contrast, our approach of optimizing for motions expressing incapability generalizes to multiple tasks, while resulting in an attempt trajectory tailored to each task.

Communicating why a robot is incapable is closely related to work that examines how robots can warn before failing. Robots can forewarn users of possible failures through text [72] or confidence levels [88, 89], trajectory timings [23], and actively choosing actions that showcase failure modes [45]. Setting accurate expectations of robot capabilities is important for narrowing the gap between the perceived and true capabilities of the robot [68].

4.3 Approach: Generating Attempt Motions

4.3.1 Expressing Incapability, Formalized

Notation

A robot's trajectory ξ is a sequence of T robot configurations: ξ_t is the configuration of the robot at time t. $\phi_b : \mathcal{Q} \mapsto \mathrm{SE}(3)$ is the forward kinematics function at body point b, and thus gives the pose (rotation and translation) of the body part at that point. $\phi'_b : \mathcal{Q} \mapsto \mathbb{R}^3$ gives the translation of body point b. The body points we consider in our particular implementation are *ee* (end-effector), *el* (elbow), *sh* (shoulder), and *ba* (base), which we found to be a reasonable discretization of the arm.

Incompletable Task Definition

A task is defined by a starting configuration q_s, along with a desired final pose x_d for the end-effector (or, in more specific instances, a desired configuration q_d). Fig. 4.1 shows an example of x_d for the task of lifting a cup. There may also be additional constraints that define the task, such as the need to keep contact with an object as the robot moves its end-effector from $\phi_{\text{ee}}(q_s)$ to x_d.

An *incompletable* task is one in which the end-effector cannot make progress beyond a certain failure point x_f. For instance, if the cup the robot tries to lift is too heavy, then x_f would be the pose of the end-effector when it first grasps the cup (because it can no longer proceed in the task from there due to the cup's weight). x_d would be vertically above x_f: the location to which, if the robot were holding on to the cup, the cup would be lifted. Fig. 4.1 also shows x_f for this example.

Expressing Incapability as an Optimization Problem

The goal of expressive incapability trajectories is to communicate *what* the robot was attempting to do and *why* it is incapable of it—in other words, its objective and the dynamics constraints, respectively. A simple approach would be to move to x_f (i.e., as far as the robot can get with the task), and stop. Prior work has suggested also repeating this motion [70, 77].

We hypothesize we can do better, by optimizing for a robot trajectory that best communicates the robot's objective and dynamics constraints to human end-users. In Chapter 2, we explained how to achieve this by using our framework for increasing transparency, and explored this problem in a simple grid world setting, in which the dynamics constraints prevent the point robot from reaching its goal location. This

results in trajectories where the robot first gets as close to the goal as possible, then backs up, and repeats this back-and-forth motion.

Taking inspiration from this, on real robots, our idea is to continue the task by executing an *attempt* trajectory past the failure point. Our insight is that we can formalize this as an optimization problem: find an attempt trajectory that maximizes similarity to the trajectory from x_f to x_d that would have been executed, had the incapability not existed. We solve this optimization subject to the dynamics constraint that the end-effector cannot proceed further.

We capture similarity, or rather dissimilarity, via a cost function $c(\xi, x_f, x_d)$, and find the attempt trajectory as:

$$\begin{aligned} \xi^* = \underset{\xi}{\text{argmin}} \quad & c(\xi; x_f, x_d) + \frac{1}{\lambda}\sum_{t=0}^{T-1} \|\xi_{t+1} - \xi_t\|^2 \\ \text{subject to} \quad & \phi_{ee}(\xi_t) = x_f, \ \forall t \in \{0..T\} \\ & \text{collision-free}(\xi). \end{aligned} \tag{4.1}$$

This objective trades off between the similarity cost and a smoothness term common in trajectory optimization [90, 91].

Cost Functions

Crucial to generating a good attempt trajectory is finding a good cost function $c(\xi; x_f, x_d)$. We investigate cost functions that seek to mimic the change from x_f to x_d. But since the end-effector cannot move in ξ, c cannot just consider the end-effector's motion: it has to consider the configuration space.

If the desired configuration q_d is not provided, we define it as the inverse kinematics solution for x_d that is closest to the starting configuration ξ_0 for the attempt:

$$\begin{aligned} q_d = \quad \underset{q}{\text{argmin}} \quad & \|q - \xi_0\|^2 \\ \text{subject to} \quad & \phi_{\text{ee}}(q) = x_d. \end{aligned} \tag{4.2}$$

Configuration-Based Cost $c_{\mathbf{q}}$: A natural starting point is to try to mimic in ξ the change in *configuration* from ξ_0 (with the end-effector at the failure point x_f) to q_d:

$$c_{\text{q}}(\xi; x_f, x_d) = d(\xi_T - \xi_0, q_d - \xi_0), \tag{4.3}$$

where d is some distance metric such as the ℓ_2-norm (we discuss options below).

Workspace-Based Cost $c_{\mathbf{b}}$: Since configuration spaces can sometimes be counter-intuitive, we also look at a cost that tries to mimic, for each body point, the change

in *position* for that body point:

$$c_{\mathrm{b}}(\xi; x_f, x_d) = \sum_{b \in B} d(\phi'_b(\xi_T) - \phi'_b(\xi_0), \phi'_b(q_d) - \phi'_b(\xi_0)) \tag{4.4}$$

Despite the end-effector staying put, this incentivizes, for instance, the elbow to move in the same direction it would have moved had the task been successful.

Emulate End-Effector Cost c_{ee}: We also introduce a third, somewhat less obvious cost function. Since the end-effector is central to the task, and now it cannot proceed further, this cost function tries to mimic using the *other* body points what the end-effector would have done:

$$c_{\mathrm{ee}}(\xi; x_f, x_d) = \sum_{b \in B} d(\phi'_b(\xi_T) - \phi'_b(\xi_0), x'_d - x'_f) \tag{4.5}$$

Distance Metrics

Each of these costs relies on a distance metric between vectors. We consider three distance metrics $d(v_1, v_2)$:

1. The squared ℓ_2-norm encourages v_1 and v_2 to have similar direction and magnitude:
$$d_{\ell 2}(v_1, v_2) = \|v_1 - v_2\|^2. \tag{4.6}$$

2. The (negative) dot product encourages v_1 and v_2 to have similar direction and large magnitudes:
$$d_{\mathrm{dot}}(v_1, v_2) = -\, v_1 \cdot v_2 = -\, \|v_1\|\, \|v_2\| \cos\theta. \tag{4.7}$$

3. We also introduce a generalization of the dot product that uses a hyperparameter k to control the trade-off between v_1 and v_2 having similar direction versus large magnitudes:
$$\begin{aligned} d_{\mathrm{proj}}(v_1, v_2; k) &= -\, v_1 \cdot v_2 \left(\frac{v_1 \cdot v_2}{\|v_1\|\, \|v_2\|} \right)^{k-1} \\ &= -\, \|v_1\|\, \|v_2\| \, (\cos\theta)^k. \end{aligned} \tag{4.8}$$

The last two distance metrics are motivated by the fact that a larger magnitude makes the attempt trajectory more obvious to human observers. Intuitively, d_{proj} projects v_1 onto v_2, projects the result back onto v_1, projects the result onto v_2, and so on. The hyperparameter k defines how many times this projection happens, so for larger k, matching direction matters more than large magnitudes. Note that $d_{\mathrm{proj}}(v_1, v_2; 1) = d_{\mathrm{dot}}(v_1, v_2)$.

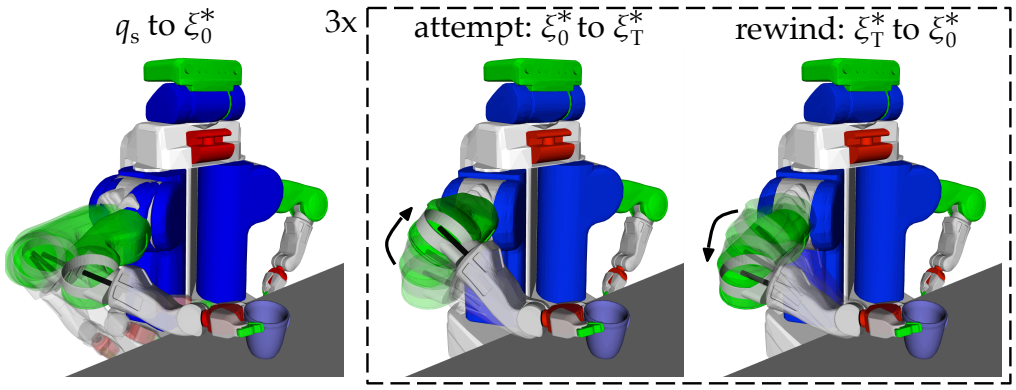


Figure 4.2: For a given incompletable task, the robot first executes the task until the point of failure (left), at which point it executes the attempt trajectory ξ^* (center). To emphasize this motion, the robot then executes the reverse of ξ^* to rewind back to ξ_0^* (right), and repeats this two more times.

Overall Attempt

The robot starts at q_s, and moves along the normal task execution trajectory to the point of failure x_f; at this point its configuration is ξ_0^*, where ξ^* is the optimum from Eqn. (4.1). From there, it executes ξ^*.

Since prior work on using motion to express incapability found repetitions to be useful [70, 77], we also explore rewinding and repeating: the robot executes the reverse of ξ^* to get back to ξ_0^*, and repeats the execute-rewind twice more, as in Fig. 4.2.

In what follows, we first show the outcome ξ^* that each cost function leads to, and use this to select a good cost function. We then run experiments to determine the appropriate relative timing of the ξ^* and the rewind (to enhance expressiveness [23]), as well as whether repetitions of the attempt help. Armed with the right general parameters, we conduct a main study across different incompletable tasks to test whether these motions, optimized to be more expressive, lead to better understanding of what task the robot is trying to do and why it will fail.

4.3.2 Comparing Cost Functions

In this section, we contrast the different behaviors produced by the cost functions and distance metrics.

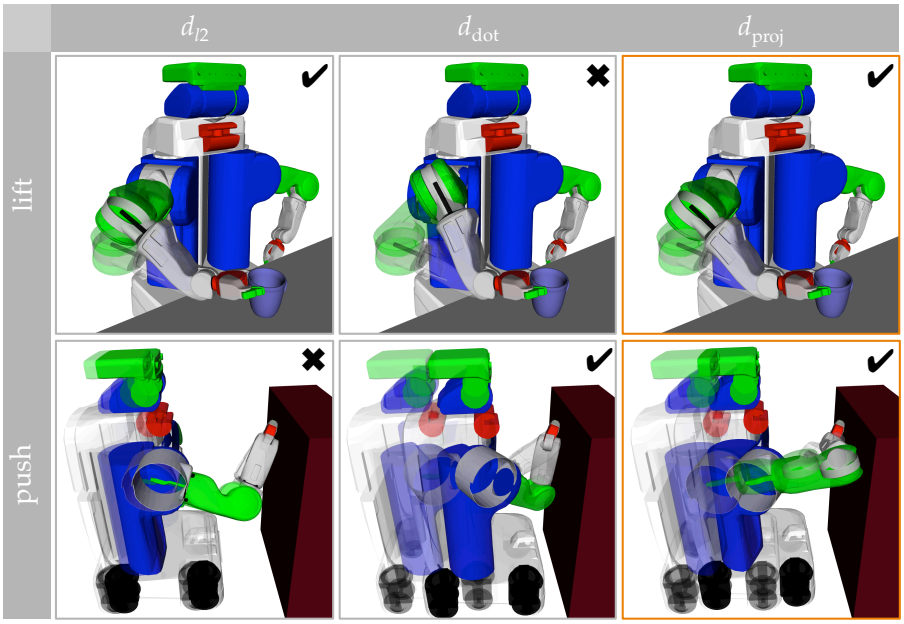


Figure 4.3: Attempt trajectories ξ^* that optimize cost function c_{ee} with each of the three proposed distance metrics. Each image shows ξ_0 (transparent) and ξ_T for that attempt trajectory. d_{proj} (last row) results in communicative attempt trajectories for both the *lift* and *push* tasks.

Implementation Details

We use a simulated PR2 robot in OpenRAVE [92] and optimize for attempt trajectories using TrajOpt [91]. Since costs c_{b} and c_{q} depend on q_d, which in turn depends on ξ_0, we simplify the optimization problem by optimizing over $\xi_{1:T}$ for each possible starting configuration ξ_0 of the attempt trajectory (found by running an inverse kinematics solver on x_f), and then select the full trajectory ξ that minimizes the objective function.

We use grid search to select the hyperparameter λ and a bias α separately for each cost function and distance metric pair.[2] We chose $k = 9$ for d_{proj}. For c_{b}, $B = \{\text{el}, \text{sh}\}$ and for c_{ee}, $B = \{\text{ba}, \text{el}, \text{sh}\}$. This is because matching the configuration would not make as much sense for the base, but the base can be useful as another body point with which to mimic the end-effector motion. In general, which body points to use might be a robot-specific question, to be determined from a few tasks

[2]We found that optimization is more stable if the terms in the objective are in the same range. So, we add a bias α to the cost function: $c'(\xi; x_f, x_d) = c(\xi; x_f, x_d) + \alpha$. We select $\lambda \in \{10, 20, 40, 80, 160\}$ and $\alpha \in \{0, 0.3, 0.6, 1.0, 2.0\}$.

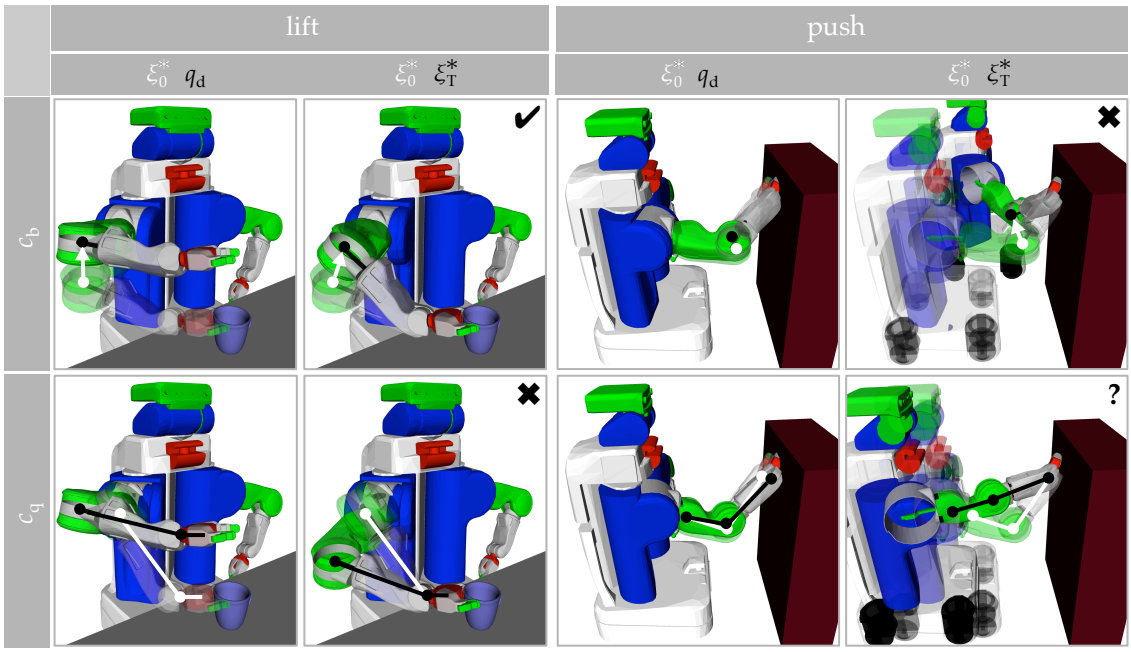


Figure 4.4: Attempt trajectories ξ^* that optimize cost function c_b or c_q, with distance metric d_{proj}. When optimizing for c_b, the attempt trajectory for *lift* is communicative, but for *push* the robot swings out to the left, which does not indicate that it is trying to push. When optimizing for c_q, the attempt trajectory for *push* is reasonable (although it could be confused for pulling, since the robot moves away from the shelf), but for *lift* the robot's elbow moves downward, which does not indicate that it is trying to lift.

and generalized to new tasks.

Behaviors

Overall, we found that c_{ee} (the cost that mimics the desired end-effector motion with the other body points) with d_{proj} (the distance metric that generalizes the dot product) is a combination that reliably leads to attempt trajectories that both move in a way that makes the task clear, and have enough movement to be noticeable. We explain this finding below by first contrasting distance metrics, and then contrasting cost functions. We use two incompletable tasks for this contrast: lifting a cup that is too heavy (the *lift* task), and pushing a shelf that is immovable (the *push* task).

Explanation of Attempt Behavior. Fig. 4.3 shows the results of c_{ee} with each distance metric. Across the board for *lift*, optimizing for c_{ee} encourages the robot to

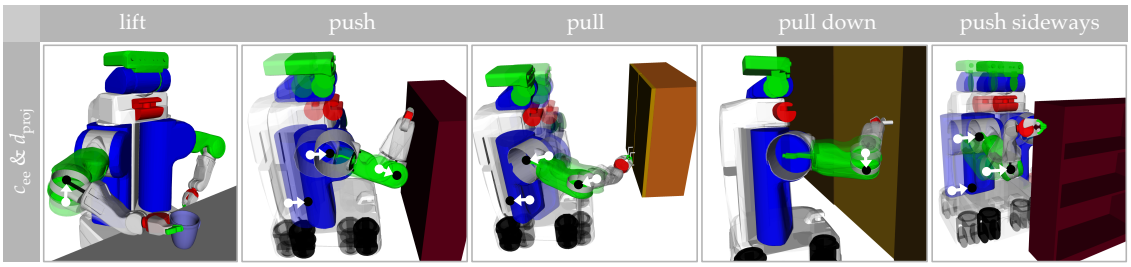


Figure 4.5: Attempt trajectories ξ^* that optimize cost function c_{ee} with distance metric d_{proj}, for five incompletable tasks. Arrows show the direction of movement for the considered body points (elbow, shoulder, and base)—for each task, these body points imitate how the robot's end-effector would move, if it were able to successfully accomplish the task.

use its elbow to produce the motion that the end-effector would otherwise produce. We thus see the robot *lifting* its elbow while keeping the end-effector on the cup that is too heavy to lift. Across the board for *push*, the robot is using its elbow, shoulder, and base, to mimic the end-effector forward motion. As a result, the robot moves *forward* toward the shelf, as the end effector stays put, unable to actually push the shelf.

Distance Comparison. d_{proj} works across both tasks. In contrast, using the $d_{\ell 2}$ distance metric results in an attempt trajectory for *push* that barely moves, and using the d_{dot} distance metric results in an over-exaggerated motion for *lift* in which the robot's elbow twists toward the center.

Cost Comparison. Now we turn to examining the performance of the other two cost functions (c_{b} and c_{q}) with the best distance metric d_{proj}.

Optimizing for c_{b} results in a confusing attempt trajectory for *push* where the robot swings to the left. This is because the elbow and shoulder body points move slightly to the left from ξ_0 to q_d: if the robot were successful in pushing, its end-effector would move further out, extending the arm, and the elbow would no longer protrude to the right, and instead move inward (to the left of the robot). The optimization thus selects an attempt trajectory that moves the elbow and shoulder as far as possible inward along this general direction (Fig. 4.4). We observe that moving the other body points (e.g., the elbow or shoulder) in the way they ideally would during a successful task execution is not always indicative of the task.

Optimizing for c_{q} results in a confusing attempt trajectory for *lift*, where the robot's elbow moves downward to match the desired configuration q_d (Fig. 4.4). In the attempt for *push*, the robot moves away from the shelf rather than toward it, also to match q_d—which would have the arm extended out after a successful push. This

could work, but could also be mistaken for pulling instead of pushing. We observe that because configuration spaces are often counterintuitive, mimicking the motion in configuration space can lead to surprising, counterintuitive motions.

In contrast, it seems that using the other body points to imitate what the end-effector cannot do might actually be indicative of what the robot is trying to achieve. We put this to the test in Sec. 4.4.3. But first, we tune the hyperparameters of attempts—the timing of the motions, and whether to include repetitions.

c_{ee} Across More Tasks. Optimizing for c_{ee} with d_{proj} also generates communicative attempt trajectories for other incompletable tasks, shown in Fig. 4.5: opening a locked cabinet (the *pull* task), turning a locked door handle (the *pull down* task), and pushing a shelf to the side (the *push sideways* task). Across all five tasks, we set $k = 3$, $\lambda = 20$, and $\alpha = 0.3$.[3] These are the attempt trajectories that we show in our user studies (in video form). A video summary of the cost comparisons and attempt motions for each task is at youtu.be/uSnUtpcdlck.

4.4 Experiments

4.4.1 Timing Motions That Express Incapability

Our aim was to manipulate timing in order to enhance the expressiveness of our optimized motions. We temporally divided a motion expressing incapability into attempt and rewind motions. The attempt motion consists of the trajectory ξ^* produced by the cost function optimization from Eqn. (4.1). The rewind motion, which is the reverse of the attempt trajectory, immediately follows the attempt motion. Our goal in this study was to find the pair of timings for the attempt and rewind motions that best convey a robot's task goal and the cause of incapability.

Experiment Design

Manipulated Variable. We manipulated timing in this study and chose three speeds (Fast, Moderate, and Slow). We were only interested in the *relative* speed between the attempt and rewind motions, so we fixed the rewind speed at Moderate and varied the attempt speed, creating three conditions: Fast attempt with Moderate rewind (Fast, Moderate), Slow attempt with Moderate rewind (Slow, Moderate), and Moderate attempt with Moderate rewind (Moderate, Moderate).

Other Variables. We tested timing across the five tasks in Fig. 4.5.

[3]To simplify optimization, we additionally assume a fixed base when computing forward kinematics for non-base body points.

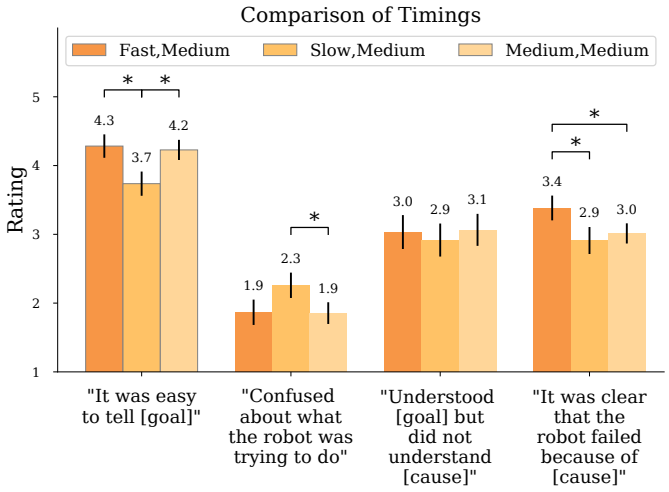


Figure 4.6: Average Likert ratings toward different timings. Timing was within-subjects, meaning participants rated each of the timing pairs. Overall, participants preferred Fast attempt and Moderate reset.

Subject Allocation. We recruited 60 participants (37% female, median age 32.5) via Amazon Mechanical Turk (AMT). All participants were from the United States and had a minimum approval rating of 95%. Timing was within-subjects: participants saw each of the three timing conditions. Task type was between-subjects: participants saw only one type of task.

Dependent Variables. Participants saw videos of all three timings. We explained to the participants what the robot was trying to do (its intended goal), and why it could not complete the task. We then asked them to help us select the timing that best expresses both the goal and cause of incapability. We created four statements to assess each timing (Fig. 4.6), and asked participants to rate their level of agreement with these statements on a 5-point Likert scale. We also asked participants to rank the three timings.

Analysis

We ran a repeated-measures ANOVA with timing as a factor and user ID as a random effect, for each item. We found significant effects of timing on how easy it was to tell the goal ($F(2, 118) = 8.26$, $p = .0004$), how confusing the goal was ($F(2, 118) = 4.47$, $p = .013$), and how clear the cause was ($F(2, 118) = 5.13$, $p = .007$). Across the board, the timing that worked best was (Fast, Moderate), as shown in Fig. 4.6. 58% of participants ranked this timing first. This indicates the attempt part

of the motion should be faster than the rewind, which intuitively makes sense—it perhaps conveys that the attempt portion is the purposeful action on which the robot expends more energy, and the rest is at a normal speed that the robot would use to move around. We use this (Fast, Moderate) timing in our main study, described in Sec. 4.4.3.

4.4.2 Comparing Repeated Attempts

After determining the best timing for the attempt and rewind motions, we looked at whether including repetition for the attempt motions enhances expressiveness.

Experiment Design

Manipulated Variable. We manipulated repetition, where we compared N=3 iterations of the attempt motion with a single (N=1) iteration of the attempt motion.
Dependent Variables. We used the same measures as in Sec. 4.4.1.
Subject Allocation. We recruited 60 participants (47% female, median age 35) via AMT. All participants were from the United States and had a minimum approval rating of 95%. Repetition type was within-subjects: every participant saw N=1 and N=3 iterations of the attempt motion. Task type was between-subjects: participants saw only one type of task.

Analysis

We ran a repeated-measures ANOVA with timing as a factor and user ID as a random effect, for each item. We found repetitions significantly increased how easy it was to tell the goal ($F(1, 58.14) = 20.21$, $p < .0001$), decreased confusion about the goal ($F(1, 62.71) = 15.95$, $p = .0002$), and made the cause more clear ($F(1, 63.64) = 16.94$, $p = .0001$). Fig. 4.7 shows the results. We proceed with repetitions for our main study.

4.4.3 Main Study: Is Expressive Motion Expressive?

With the details of how to generate motions expressing incapability out of the way, we now turn to how much this helps people identify the robot's goal and cause of incapability.

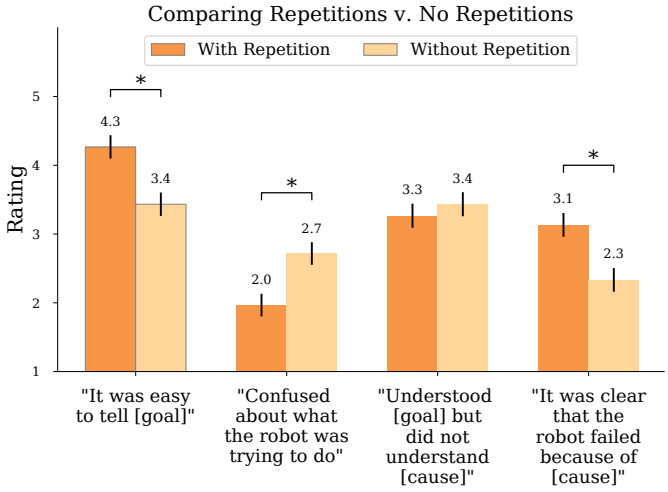


Figure 4.7: Average Likert ratings comparing repetitions with no repetitions. The study was within-subjects, meaning participants rated both motions with repetition and motions without repetition. Overall, participants preferred motions with repetitions.

Experiment Design

Manipulated Variable. We compared our motions expressing incapability against the state-of-the-art approach for automatically generating motion to express robot incapability [70,77]. With the state-of-the-art approach, the robot repeatedly executes a failing action. The manipulated variable was generating the attempt motion via our optimization-based approach versus via the repeated-failures approach.

We created motions expressing incapability following the process in Fig. 4.2. For each task, we optimize c_{ee} with d_{proj} to generate the attempt ξ^*. The robot moves from q_s to ξ_0^*, executes the attempt at speed=Fast, rewinds back to ξ_0^* at speed=Moderate, and repeats the attempt-rewind two more times for a total of N=3 iterations.

For the repeated-failure motions, the robot moves from q_s to ξ_0^*, rewinds back T time steps at speed=Moderate, and moves back to ξ_0^* at speed=Fast. The robot rewinds T time steps and moves back to ξ_0^* two more times for a total of N=3 iterations. The timing and number of iterations are the same as for our approach, to limit possible confounds.

Dependent Variables. Our dependent variables included how well participants could infer the robot's goal and cause of incapability, as well as measures regarding their perception of the robot.

We assessed ***goal recognition***—how well participants could infer the intended task goal—in several ways. First, using an open-ended response, we asked participants to state what they thought the task goal was. Second, we presented four plausible task goals, with the correct task goal as once of the choices. We then asked participants to rate, on a 5-point Likert scale labeled "Strongly Disagree" to "Strongly Agree," how well each task goal described what the robot's goal was. Lastly, we asked participants to explicitly rank the task goals in order of how well they described the robot's goal. Incorrect goal alternatives are described in Table 4.1.

We measured ***cause of incapability recognition***—how well participants infer the incapability underlying the robot's failure—in a similar way. The only difference was that the Likert-scale questions included a second correct option ("The robot was not strong enough [to complete the task]") because it is a plausible interpretation of our motion: our method is not meant to differentiate between, for instance, the cup being too heavy and the robot not being strong enough. Rather, our method is meant to differentiate between these two correct options and other causes of incapability that are possible, but untrue, for instance the robot running out of battery, its planning algorithm or software system getting stuck or crashing, and so forth, see Table 4.1.

For assessing task goal and cause of incapability, participants saw either the motions expressing incapability or the repeated-failure motions. Next we assessed participants' subjective perceptions and attitudes toward the robot, and for that, we wanted participants to compare the two "robots"—with expressive motions generated by either our optimization-based approach or the repeated failures. We felt comparisons were important here in order to ground participants' perceptions, thus improving experimental reliability. We thus introduced the other robot and asked them, for each robot, to rate their level of agreement with the statements in Fig. 4.10.

	Incorrect Goal Recognition Statements
Lift	The robot was trying to push the cup. The robot was trying to pull the cup. The robot was trying to knock over the cup.
Pull	The robot was trying to slide the cabinet door sideways. The robot was trying to sense the cabinet handle. The robot was trying to push the cabinet away.
Pull Down	The robot was trying to sense the door handle. The robot was trying to prevent someone from opening the door on the other side. The robot was trying to remove the door handle.
Push	The robot was trying to sense the box. The robot was trying to stroke the box. The robot was trying to knock on the box.
Push Sideways	The robot was trying to sense the shelf. The robot was trying to lift the shelf. The robot was trying to knock on the shelf.

Incorrect Cause of Incapability Statements
The robot had a mechanical failure (e.g. ran out of battery, arm got stuck, etc.) or software crash.
The robot did not know how to [goal].
The robot is waiting for permission to [goal].

Table 4.1: List of incorrect plausible goal and cause of incapability statements participants had to choose from. The cause of incapability statements were similar across tasks.

Subject Allocation. We recruited 120 participants (38% female, median age 33) through Amazon Mechanical Turk. All participants were from the United States and had a minimum approval rating of 95%. The optimization-based versus repeated-failure manipulation was between-subjects for the first part of the study and was within-subjects for the last part, in which we evaluated subjective perceptions of the robot. We had 24 participants for each of the five tasks, where 12 were in

the optimization-based expressive motion condition and the other 12 were in the repeated-failure condition.

Hypotheses. We hypothesized that motions expressing incapability will help participants understand the robot's goal and incapability better than repeated-failures will, across all tasks. We also hypothesized that participants will perceive the robot with motions expressing incapability more positively than they perceive the robot with repeated-failures.

H1: Motions expressing incapability improve goal recognition.

H2: Motions expressing incapability improve cause of incapability recognition.

H3: Participants perceive the robot more positively when it uses motions expressing incapability on an incompletable task.

Analysis

Goal Recognition. We first analyzed participants' ratings of the different possible goals. We ran a two-way ANOVA with motion type as the independent variable for each possible goal's rating. We found that motions expressing incapability significantly improved the rating of the correct goal ($F(1, 119) = 19.43$, $p < .0001$), and significantly decreased the average rating given to the incorrect goals ($F(1, 119) = 23.79$, $p < .0001$). This supports our hypothesis **H1**.

Fig. 4.8 shows a task breakdown of the correct goal rating. We can see that the largest improvements are in *push* and *push sideways*, probably because the context is not enough for these tasks to be conveyed by the non-expressive motion—some attempt is really needed to understand what the robot is trying to do. In contrast, as the robot reaches for the cup, its intended goal of lifting the cup becomes pretty clear even without an attempt.

Next, we analyzed participant's rankings of the possible goals. Motions expressing incapability significantly improved the ranking of the correct goal ($F(1, 119) = 30.69$, $p < .0001$) from an average ranking of 1.65 (already close to the top) to one of 1.05, with nearly all participants selecting the correct goal (95% as opposed to 60%).

We also analyzed participants' open-ended responses. We categorized an open-ended response as correct if the response contained all keywords, or synonyms of keywords, from the correct goal recognition statement. For example, in the correct statement, "The robot was trying to pick up the cup," we designated "pick up" and "cup" as keywords. We had two experimenters code the statements where one coder coded all statements and the other coded 10%. There was strong agreement between the two coders' judgments, with Cohen's $\kappa = 1$, $p = .001$. We found that participants who saw the motion expressing incapability were able to explain the robot's goal correctly (in their open-ended response) significantly more than participants who

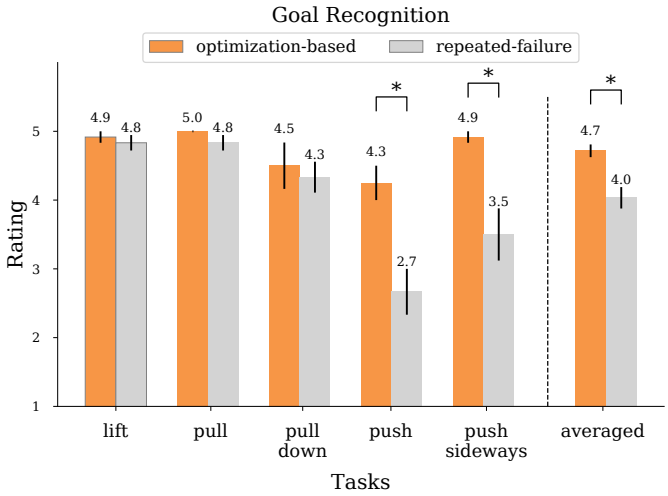


Figure 4.8: Ratings toward the correct goal for each task. A higher value indicates higher confidence. The averaged values represent mean ratings of expressive and non-expressive motions across tasks.

saw the repeated-failure, $p = .0003$.

Finally, we analyzed the two goal-recognition Likert-scale subjective questions at the end of the study ("It was easier to tell [the robot's goal]." and "I was confused about what the robot was trying to do."). We used a repeated-measures ANOVA, since this part was within-subjects. We found that motions expressing incapability improved the ease of identifying the goal ($F(1, 119) = 602.38$, $p < .0001$) and decreased confusion about the goal ($F(1, 119) = 183.13$, $p < .0001$).

> *Overall, our results support **H1**: our motions expressing incapability improved goal recognition.*

Cause of Incapability Recognition. Looking first at the ratings for possible causes of incapability, participants rated five causes, where one was the robot's actual cause of incapability, and another was that the robot was not strong enough—a plausible cause that is hard to disambiguate from the actual cause within each task. The other causes were incorrect.

We found that motions expressing incapability significantly improved the rating of the correct cause ($F(1, 119) = 13.6$, $p = .0003$), but did not have a significant effect on the rating of the plausible cause. Motions expressing incapability decreased the average ratings of the incorrect causes ($F(1, 114) = 19.05$, $p < .0001$). Fig. 4.9 shows a task breakdown of the correct cause of incapability rating. We see that there is an improvement across all tasks, with the largest improvement in *push*.

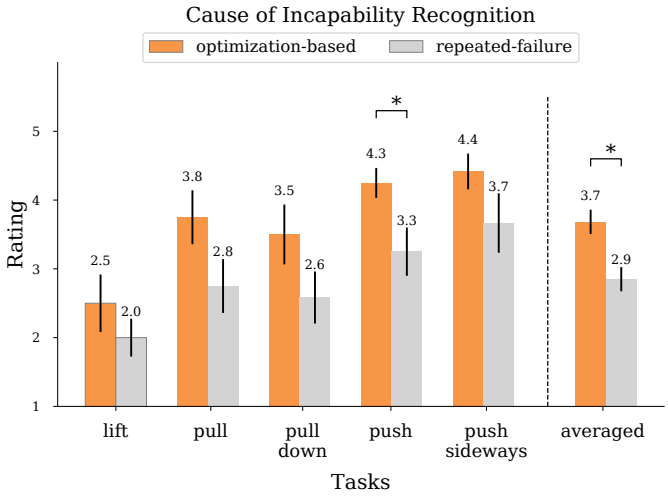


Figure 4.9: Ratings toward the correct cause of incapability for each task. A higher value indicates higher confidence. The averaged values represent mean ratings of optimization-based and repeated-failure motions across tasks.

Motions expressing incapability also significantly improved the rank of the correct cause ($F(1, 119) = 23.71$, $p < .0001$), from an average of 3.02 to an average of 1.98.

Next, we looked at participants' open-ended responses. We used the same coding scheme as we did for the goal-recognition open-ended statements. We found that there was a strong agreement between the two coders' judgments, with Cohen's $\kappa = 1$, $p = .001$. Participants who saw the motions expressing incapability were significantly more likely to describe the correct cause of incapability compared to those who saw the repeated-failures, $p = .0002$.

Finally, we conducted a repeated-measures ANOVA on the one subjective rating relevant to cause of incapability recognition ("It was clear that [cause of incapability]."). We found motions expressing incapability significantly improved this rating ($F(1, 119) = 182.31$, $p < .0001$).

> *Overall, our results support **H2**: our method for generating motions expressing incapability improved cause of incapability recognition.*

Perception of Robot. We ran a repeated-measures ANOVA for each statement. With motions expressing incapability, users perceived the robot as more like an animated character ($F(1, 119) = 30.82$, $p < .0001$), wanted to help the robot more ($F(1, 119) = 51.93$, $p < .0001$), thought it was more trustworthy ($F(1, 119) = 31.76$, $p < .0001$) and a better teammate ($F(1, 119) = 85.97$, $p < .0001$), and were more

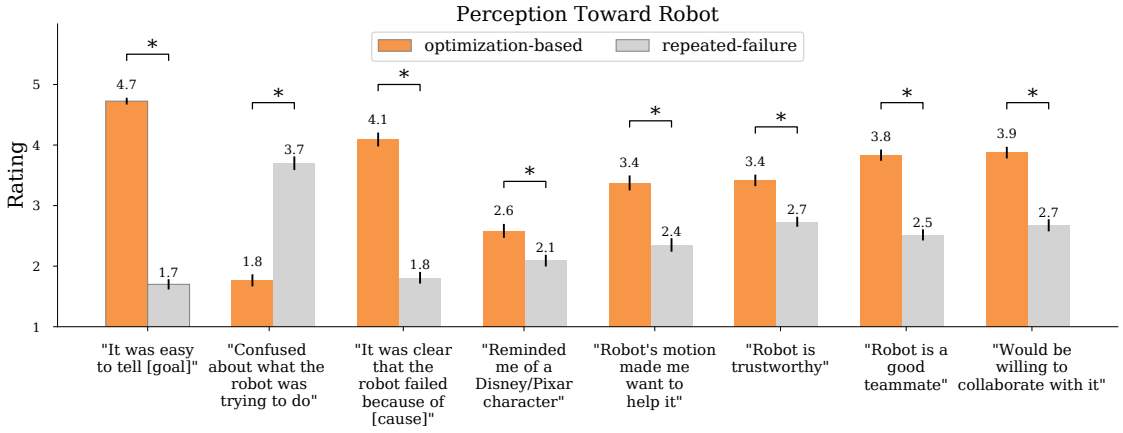


Figure 4.10: Ratings toward the Likert statements used in Sec. 4.4.3. A higher value indicates higher confidence. For each statement, ratings for the optimization-based and repeated-failure conditions were averaged across tasks.

willing to collaborate with the robot in the future ($F(1, 119) = 69.66$, $p < .0001$). See Fig. 4.10 for details.

> *Overall, our results support **H3**: our method for generating motions expressing incapability improved users' perceptions of the robot.*

4.5 Discussion

Summary

We use an optimization-based approach to automatically generate expressive trajectories that communicate *what* a robot is trying to do and *why* it will fail. The optimization produces a trajectory where body points on the robot "mimic" how the end-effector would move if the robot had been capable of completing the task. We complemented the expressiveness of our optimized trajectory by manipulating repetition and timing. Our results show that, compared to the state-of-the-art approach, motions expressing incapability improve intent recognition and cause of incapability inference while also increasing positive evaluations of the robot.

Our optimization enables robots to automatically and efficiently generate motions expressing incapability. On average, solving Eqn. (4.1) for for cost c_{ee}, distance d_{proj}, and a specified q_f takes half a second when the base does not move (e.g., for the *lift* and *pull down* tasks) and a few seconds when the base is able to move. Given

this, it is feasible for the robot to detect incapability in the middle of execution, and compute an expressive motion on the fly.

Limitations and Future Work

Perhaps the greatest limitation of our work is that our optimization covers only a narrow set of tasks the robot is incapable of completing. Incapabilities that are not about the end-effector position changing, such as grasping, or incapabilities that have nothing to do with the end-effector, such as those related to perception, cannot be communicated using our optimization. Our approach also does not address when or how expressive motions should be accompanied by other channels of communication, such as verbal communication. Although these are very important areas for future work, we are excited to see that the same method can *automatically* generate motion that is expressive and useful *across a range* of different tasks.

Chapter 5

Establishing Appropriate Trust in Black-Box Policies

Learned neural network policies make it particularly challenging for humans to have an accurate mental model of a robot's capabilities and how it acts. In this chapter, we propose an approach for helping end-users build a mental model of such policies. Our key observation is that for most tasks, the essence of the policy is captured in a few *critical states*: states in which it is very important to take a certain action. Our user studies show that if the robot shows a human what its understanding of the task's critical states is, then the human can make a more informed decision about whether to deploy the policy, and if she does deploy it, when she needs to take control from it at execution time.[1]

5.1 Motivation and Background

When humans have an accurate mental model of a robot, their subsequent interactions with this robot are safer and more seamless. This mental model may include the robot's intentions [20, 43, 44], its objectives (as explored in Chapter 3), its capabilities [45] (and further explored in Chapter 4), or its decision-making process [93].

In particular, giving human end-users an accurate mental model of a robot's capabilities is key to establishing an appropriate level of trust in the robot [94–96]. Establishing *appropriate* levels of trust in robots is essential: if end-users do not trust a robot, they may unnecessarily interfere with its operation, and will fail to take advantage of all its capabilities [97]. On the other hand, if end-users over-trust

[1] This work was published as *Establishing appropriate trust via critical states* in IROS 2018 [15].

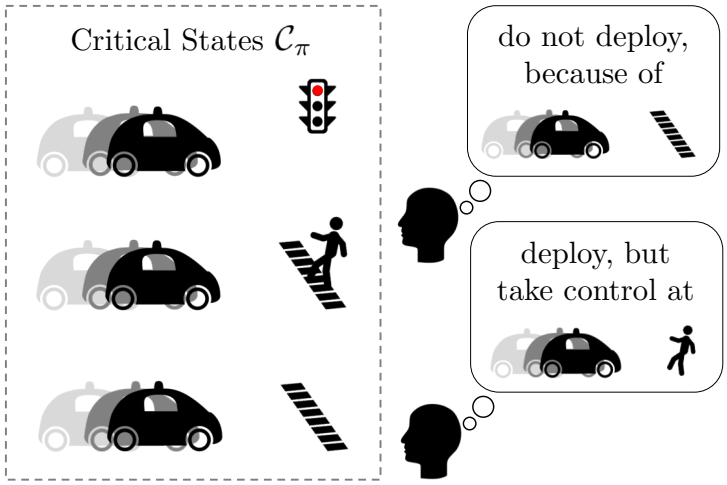


Figure 5.1: By introducing human end-users to a policy π's critical states $\mathcal{C}_\pi$ (left), we enable them to make a more informed decision about whether to deploy the policy, and when to take control from it. For example, suppose that a self-driving car's policy believes it is critical to stop when it encounters red lights, a pedestrian crossing a crosswalk, and an empty crosswalk. An end-user (top right) might see these critical states and decide not to ride in this car, because the last critical state is clearly incorrect. A different end-user (bottom right) might be comfortable riding in this car, but will be more aware of possibly needing to take control when there is a pedestrian crossing the road without a crosswalk in sight, because the policy did not consider that to be a critical state.

a robot, they will expect it to act correctly in situations that it in fact cannot handle, which leads to unexpected behavior, and perhaps injuries and damage. As robots become more capable, they may unintentionally lead humans to over-trust them [68]. In general, trust is a complex phenomenon, and there are a variety of ways in which robots and machines may influence human end-users' trust [98–100].

Forming an accurate mental model, and thus establishing appropriate trust, is particularly challenging when the robot has learned a complex black-box policy. For instance, recently neural network policies have been trained to perform robotic manipulation skills [101] and drive in the real world [102]. These neural networks are trained end-to-end to map directly from raw inputs (e.g., images) to a distribution over actions to take. To decide how much to trust a learned policy, we have to know whether the robot has figured out the correct actions to take. But, it is impossible to examine what the robot plans to do in every possible state.

Our insight is that the end-user does not need to know what the robot would do in *all* states. For many tasks, in most states the ultimate outcome of the task is similar,

regardless of which action the robot takes locally. But there are a few states—*critical states*—where it really matters which action the robot takes. For instance, imagine an autonomous car driving down a highway. When there are no vehicles nearby, it does not matter whether the car maintains its current speed, speeds up or slows down slightly, or turns slightly to the right or left. In contrast, if the vehicle directly in front slams on its brakes, the autonomous car must immediately slow down as well. The latter is a critical state, whereas the former is not.

If a robot's policy is stochastic, then it should output a low-entropy action distribution for critical states. A (stochastic) policy's critical states are thus a concise summary of that policy, since in all other states, the policy has no strong preference about which action to take. So, showing how a policy acts in critical states is the optimal thing to do from the standpoint of increasing transparency to human end-users (as discussed in Chapter 2).

Motivated by this, we propose showing end-users how a robot acts in critical states, to give them a better understanding of what it has learned, and enable them to decide which situations to trust the robot in (Fig. 5.1). After seeing how a robot acts in critical states, a potential user may decide that this robot is not trustworthy, and *decline to use it*. Or, in human-in-the-loop setups—for instance, a passenger riding in a self-driving car, or an engineer supervising robot arms in a factory—this ensures users are well-equipped to decide *when they need to take control* over the robot's operation.

Our main contribution is a method for *algorithmic assurance* [103], that enables end-users to more quickly establish an appropriate level of trust in robots that they interact with, rely on, or supervise. Our user studies suggest that humans are indeed able to develop more appropriate trust in a robot through observing how it acts in what it considers to be critical states, compared to just observing it act over time. We evaluate this through both self-reported measures of trust, as well as through allowing users to take control during execution of the policy [97]: if they have developed an appropriate level of trust, they would only choose to take control in critical states that the robot likely cannot handle.

5.2 Approach: Computing & Using Critical States

5.2.1 Preliminaries

Notation

We consider the setting of a Markov Decision Process (MDP), defined by $\{\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma\}$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ the action space, $\mathcal{P} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$

the transition probabilities, $\mathcal{R} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ the reward function, and $\gamma \in (0, 1]$ the discount factor.

A robot's policy π is a stochastic function mapping each state to a distribution over actions ($\pi : \mathcal{S} \to \Delta_{\mathcal{A}}$, where $\Delta_{\mathcal{A}}$ is the probability simplex on $\mathcal{A}$). Its value function at state s is

$$V^{\pi}(s) = \max_{a} \int_{s'} P(s, a, s')[R(s, a, s') + \gamma V^{\pi}(s')], \tag{5.1}$$

and its action-value function at state s and taking action a is

$$Q^{\pi}(s, a) = \int_{s'} P(s, a, s')[R(s, a, s') + \gamma \max_{a'} Q^{\pi}(s', a')]. \tag{5.2}$$

In this framework, a critical state s is one for which $Q^{\pi}(s, a)$ varies greatly across different actions a: there are a small number of actions for which $Q^{\pi}(s, \cdot)$ is high, but for most actions it is mediocre or low. We will define this formally in the next section.

Maximum-Entropy Reinforcement Learning

Typically a robot's goal is to maximize expected cumulative discounted reward, or return:

$$\mathbb{E}_{\pi, \mathcal{P}} \left[\sum_{t} \gamma^{t} R(s_t, a_t, s_{t+1}) \right]. \tag{5.3}$$

Depending on the MDP, this may result in policies that are essentially deterministic, treating all states as critical.

In contrast, in maximum-entropy reinforcement learning, the policy is trained to not only maximize return, but also to act as randomly as possible while doing so [57, 104, 105]. Concretely, the policy is trained to maximize

$$\mathbb{E}_{\pi, \mathcal{P}} \left[\sum_{t} \gamma^{t} [R(s_t, a_t, s_{t+1}) + \alpha \mathcal{H}(\pi(\cdot|s_t))] \right], \tag{5.4}$$

where α determines the tradeoff between maximizing return and entropy, and $\mathcal{H}(\pi(\cdot|s_t))$ is the entropy of the policy's output action distribution at state s_t. This leads to a policy with meaningful critical states, since it learns to acts randomly in states where the action has little impact on return, and to act purposefully in states where the action does have a major impact on return.

We train our neural network policies using Soft Actor-Critic[2] (SAC) [105], a deep reinforcement learning method that is based on maximum entropy reinforcement learning. We find that in practice, training with SAC indeed produces policies with meaningful critical states.

5.2.2 Computation of Critical States

Policy-Based

Recall that critical states are those in which a policy (or human) greatly prefers a small set of possible actions over all others. A natural definition of the set of critical states $\mathcal{C}_\pi$ for a stochastic policy π is thus

$$\mathcal{C}_\pi = \{s \,|\, \mathcal{H}(\pi(\cdot|s)) < t\}, \tag{5.5}$$

where $\mathcal{H}(\pi(\cdot|s))$ is the entropy of the policy's output action distribution at state s, and $t \in \mathbb{R}$ is the threshold for being considered "critical." This definition of critical states can be applied to both continuous and discrete action spaces.

Value-Based

Certain reinforcement learning approaches for training policies, such as actor-critic methods, also learn a value or action-value function in parallel to (or instead of) learning a policy [106]. Value functions capture the long-term consequences of a policy's actions, so when they are available, they are a reasonable alternative for computing critical states.

If we define critical states more concretely as those in which acting randomly will produce a much worse result than acting optimally, then the set of critical states $\mathcal{C}_\pi$ for a stochastic policy π is:

$$\mathcal{C}_\pi = \{s \,|\, (\max_a Q^\pi(s,a) - \frac{1}{|\mathcal{A}|}\sum_a Q^\pi(s,a)) > t\}, \tag{5.6}$$

where Q^π is the learned action-value function. If the action space is continuous, this can be applied after discretization. Computing critical states based on a learned value function V^π is also possible, by using one-step rollouts to estimate Q^π for each action.

We train our policies with SAC, which learns a policy and an action-value function in parallel. In practice, we found that computing critical states based on action-value

[2]We use the implementation at github.com/haarnoja/sac.

functions was more reliable, because the policy may learn to exploit environment characteristics (e.g., action clipping) to maximize entropy.

Note that with either of these two approaches, computing the critical states of a policy is agnostic to the implementation of the policy itself; only access to either the policy's or action-value function's output is required, so this can be directly applied to black-box policies.

5.2.3 Using Critical States

We assume a human expert at the task. Let $\mathcal{C}_h$ be the set of (ground-truth) states that she considers critical. We do not know what $\mathcal{C}_h$ is—and, in fact, this may differ across human end-users—so we cannot check whether $\mathcal{C}_\pi$ and $\mathcal{C}_h$ are the same. However, what we can do is expose the human to $\mathcal{C}_\pi$. Below, we describe the interaction we envision.

Decline to deploy due to false positives, false negatives, incorrect actions

Before using a robot that has learned a policy π, the human end-user first gets to observe its actions in the states *it* considers as critical, $\mathcal{C}_\pi$. If the human spots false-positive or false-negative critical states (i.e., states that are in $\mathcal{C}_\pi$ but not in $\mathcal{C}_h$ or vice versa), then she can decline to deploy the robot. False-negative critical states happen, for instance, when an autonomous car does not realize that stopping for a red light is a critical state. False-positive critical states happen, for instance, when an autonomous car considers it critical to slow down, even if there is quite a bit of space left to the car in front. Both false-negative and false-positive critical states indicate that the robot has failed to learn something fundamental about the task, and thus perhaps should not be trusted. Similarly, if the policy identifies a true-positive critical state but is mistaken about which action is correct in that state, then the end-user will observe that and not trust the policy as a result.

Take control

We are also interested in the case where $\mathcal{C}_\pi$ does not have any *obvious* false-positive, false-negative, or incorrect-action critical states, and the user decides to go ahead and deploy the robot, but the robot operates with the user in the loop. At execution time the user is able to take control from the policy whenever she deems it necessary. Because she has already observed how the policy acts for states in $\mathcal{C}_\pi$, the user is better equipped to take control from the policy when necessary, and refrain from doing so when not necessary.

5.2.4 Justification of Critical States

The user should have enough information based on critical states to take control when necessary at execution time. Note that at execution time, any state s encountered by the robot must fall into one of three cases: (1) $s \notin \mathcal{C}_h$, (2) $s \in \mathcal{C}_h$ and $s \in \mathcal{C}_\pi$, or (3) $s \in \mathcal{C}_h$ and $s \notin \mathcal{C}_\pi$.

In case (1), the user does not consider this state to be critical, so by definition she does not care which action the policy chooses and will refrain from taking control. In contrast, in cases (2) and (3), the user does consider this state to be critical, and cares about which action the policy takes. Since the user has observed (and approved) the policy's actions for states in $\mathcal{C}_\pi$, she should trust the robot in case (2). In case (3), s is a false-negative critical state that the end-user forgot about when approving this policy. Since this is a critical state that the policy does not know is critical, she should take control from the policy immediately.

If the user had not been able to observe how the policy acts for states in $\mathcal{C}_\pi$, then she would not be able to distinguish between when she absolutely must take control (states in case (3)), and when she should not but may be tempted to (states in case (2)).

5.3 Experiments

5.3.1 User Study: Impact of Critical States

We begin by investigating how human end-users draw conclusions after observing the critical states of a policy, and how they respond to different errors (i.e., false positives, false negatives, or incorrect actions) in these critical states. In order to explore this in a systematic way, instead of obtaining critical states from trained policies, we construct sets of critical states where each set has at most one error. From this we can learn, for example, how much seeing a false-positive critical state impacts trust, versus seeing an incorrect-action. Later, in our main user study (Sec. 5.3.2), we expose end-users to critical states from actual trained policies.

Experiment Design

The study consists of three phases. In the *query* phase, we first introduce participants to the task and ask them, for a handful of states, whether they consider it critical to take a particular action in that state (Fig. 5.2, top row). This is to get a sense of what $\mathcal{C}_h$ is across participants. In the *exposure* phase, we introduce participants to a policy, for instance by showing them its critical states. Finally, in

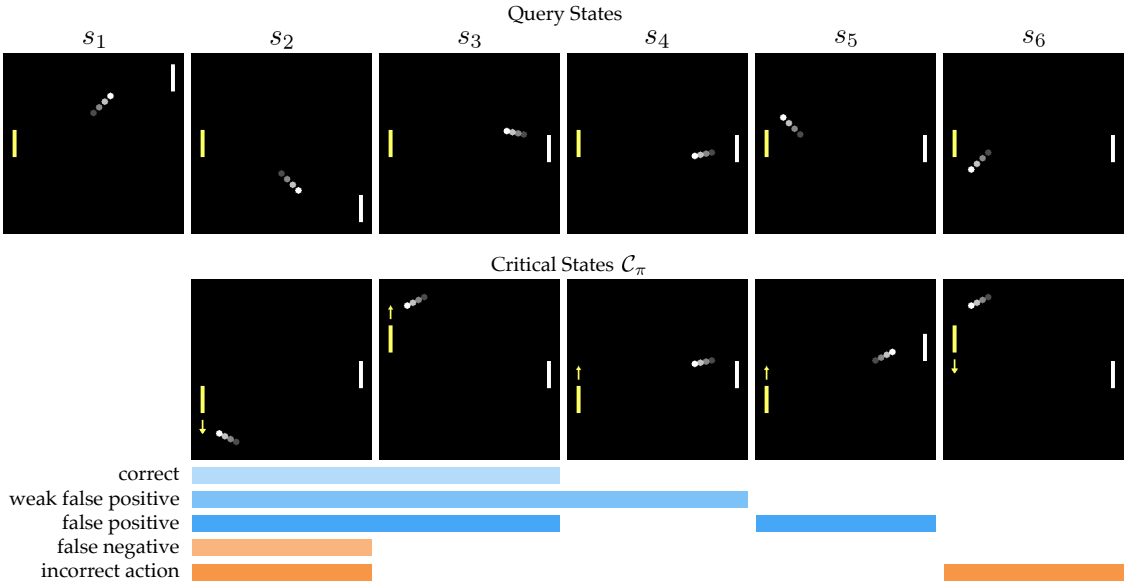


Figure 5.2: The query states and sets of critical states $\mathcal{C}_\pi$ shown in our user study for Pong. The policy controls the yellow paddle. Query states s_1 through s_4 are not critical, because the paddle has plenty of time to reach the ball, whereas s_5 and s_6 are. The colored bars indicate which states are included in each possible $\mathcal{C}_\pi$. For example, the *incorrect-action* $\mathcal{C}_\pi$ contains one correct critical state (the leftmost one) and one incorrect-action critical state (the rightmost one). The *false-negative* $\mathcal{C}_\pi$ contains one correct critical state, but is missing the second correct critical state—so the corresponding policy would likely miss balls heading toward it from above. Each possible $\mathcal{C}_\pi$ contains at least one correct critical state (the leftmost one).

the *test* phase, we ask participants whether they would take control from the policy, for each of the same states as in the *query* phase.

Domain. We chose a straightforward task with clear critical states: Pong. In Pong, a ball bounces back and forth between two paddles, and the goal is to use your paddle to hit the ball past your opponent's. So states in which the ball is headed back toward your opponent are non-critical, since it does not matter much how you move your paddle. In contrast, states in which the ball is heading toward your paddle and has almost reached it are critical.

Manipulated Variables. We manipulate the set of critical states $\mathcal{C}_\pi$ shown to the participant. We construct five options for $\mathcal{C}_\pi$—*correct*, *false-negative*, *weak-false-positive*, *false-positive*, and *incorrect-action*—that cover all the possible problems with a particular policy's critical states (Fig. 5.2). In the baseline condition, instead of showing the participant a set of critical states, we simply give them a summary

statistic of the robot's performance: "this policy wins in 95% of cases." This establishes a baseline of how much participants trust policies for Pong that are reasonably good.

Dependent Measures. We are interested in whether observing a set of critical states leads participants to develop *appropriate* trust in the policy that generated those critical states. We measure trust in two ways: subjectively with five-point Likert questions, and objectively with which *test* phase states participants choose to take control from the policy in, and whether those are correct (i.e., in C_h and either not in C_π, or in C_π but as an incorrect action). This *test* phase simulates execution-time: after the end-user has already chosen to deploy the policy, and is now supervising it.

Hypothesis.

H1. When $\mathcal{C}_\pi$ contains false-negative, false-positive, or incorrect-action critical states, users are less inclined to trust the policy π, compared to if its critical states match $\mathcal{C}_h$ perfectly (i.e., the *correct* condition).

H2. In states that are critical (i.e., in $\mathcal{C}_h$), participants will take control if a policy π's critical states $\mathcal{C}_\pi$ suggest that this policy will not choose the correct action in this state. For example, since the *false-negative* $\mathcal{C}_\pi$ for Pong is missing critical states in which the paddle needs to immediately move upward to hit the ball, this should lead participants to take control in similar states at execution time (e.g., query state s_5). But, they should not take control at state s_6, since the *false-negative* $\mathcal{C}_\pi$ includes a similar critical state and chooses the right action.

Subject Allocation. We used a between-subjects design. We ran this experiment on a total of 72 participants across the six conditions, recruited via Amazon Mechanical Turk. The average age of the participants was 31.4 ($SD = 6.7$). The gender ratio was 0.32 female.

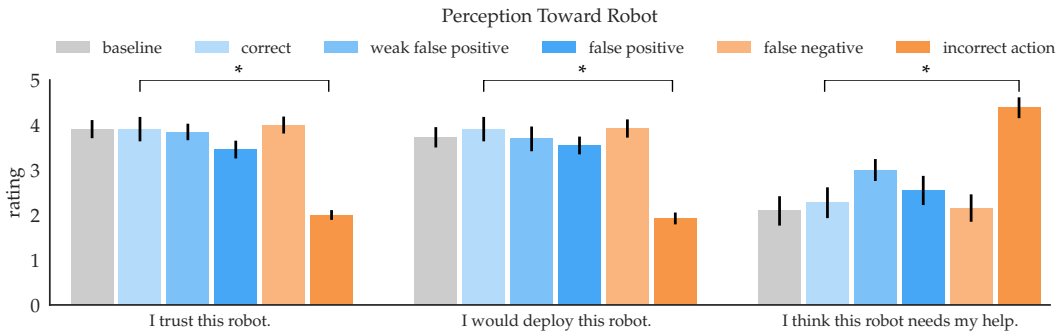


Figure 5.3: Ratings for Likert statements in Sec. 5.3.1, averaged across participants in each condition. Higher ratings mean higher agreement.

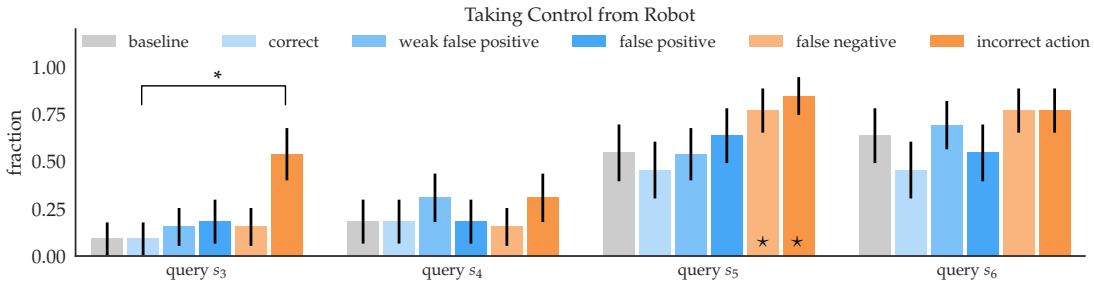


Figure 5.4: Participants' yes/no responses for whether they would take control of the policy at a particular query state (from Fig. 5.2). A $\star$ indicates that this is a state in which participants should choose to take control, based on the critical states they observed. Results for s_1 and s_2 are omitted—people overwhelmingly chose to not take control, regardless of which condition they were in.

Analysis

Subjective. We asked participants how much they trust the robot, whether they would deploy it, and whether they thought the robot needed their help (Fig. 5.3).

We found a significant difference between *incorrect-action* and *correct* for all three subjective measures (Student's t test, $p < 0.0001$). However, *false-positives* and *false-negatives* did not decrease users' perception compared to *correct* (the trend is in the right direction for the false positives). This may be because Pong is a relatively simple domain, which makes humans more inclined to give policies the benefit of the doubt, in terms of being able to generalize to other critical states (in the case of the *false-negative* $\mathcal{C}_\pi$).

Objective. We also asked participants, for each of the six query states (Fig. 5.2), a yes/no question for whether they would take control of the policy at that state (Fig. 5.4). In the *query* phase, participants agreed that of the six states, only s_5 and s_6 are truly critical (i.e., in $\mathcal{C}_h$). We see that overall, across all conditions, participants tend to take control in these two critical states, and not in the others. This supports our assumption that humans will tend to only take control of policies in states that are within $\mathcal{C}_h$.

However, this also indicates that participants are taking control even when it is not necessary. For instance, users who saw the *correct* $\mathcal{C}_\pi$ saw it act correctly in states similar to both critical query states, but still almost half of users choose to take control in that state.

On the bright side, we saw a number of trends in line with our hypothesis. First, we do notice that for these two critical query states, users tend to be less likely to take control after seeing *correct* $\mathcal{C}_\pi$, compared to just being told a summary statistic

about the policy, in the baseline condition.

Second, participants in the *incorrect-action* condition again indicated low trust in the robot, by choosing to take control more often, even in state s_3, which is only weakly critical. We found participants chose to take control significantly more in the *incorrect-action* condition than the *correct* condition for s_3 (Student's t, $p < 0.01$) and s_5 ($p = 0.05$).

Third, participants who saw false-positive and false-negative critical states actually tended to take control more often than those who saw correct ones, suggesting that they did pick up somewhat on the problems indicated by $\mathcal{C}_\pi$ (with weak significance, for s_5 and s_6, $p = 0.11$).

Summary. Overall, participants responded most strongly to critical states that reveal incorrect actions. There, they would intervene before deployment. For false negatives, they would tend to take control away from the robot more compared to participants who saw correct critical states. False positives only benefited from slight improvements in how much participants would take control, though at the same time false positives are the smallest of errors, as we discussed in Sec. 5.2.

5.3.2 User Study: Utility of Critical States

Our previous study analyzed how people respond to different errors that critical states might reveal. In our main user study, we evaluate the utility of showing the critical states of a policy π against other options of exposing end-users to the policy, in terms of establishing appropriate trust.

We train two neural network policies for a driving domain, and hypothesize that critical states are best at helping people figure out which one is better. We train these policies using SAC, and use Gaussian mixture policies with four components [105].

In practice, critical states in $\mathcal{C}_\pi$ may be very similar to each other, so instead of showing all states in $\mathcal{C}_\pi$ to the human, we first cluster these states (with k-means++) and then show the policy's behavior in the most critical state from each cluster. We take advantage of the fact that neural network policies learn hidden-layer feature representations, and use the output of the last hidden layer as features for clustering. Concretely, we collect 10,000 timesteps by rolling out each policy, cluster the 10% most critical states into ten clusters, and show the most critical state from each cluster. So, we end up showing ten critical states per policy.

Experimental Design

This study consists of the same three phases as the previous study.

Domain. We train policies to drive in a top-down driving simulator that mimics highway driving. The goal of the policy is to navigate down this road while passing other, slower cars. Car dynamics follow the bicycle vehicle model [64]. The state space consists of an indicator for which lane the robot car is currently in, its position and heading, and the relative positions, heading, and speed of other nearby cars. The action space is continuous and one-dimensional, in the range $[-1, 1]$; it corresponds to the change in steering angle.[3] The reward function encourages forward progress and penalizes getting close to other cars, being off-center in the lane, turning, and steering sharply.

Manipulated Variables. We manipulate two variables: how a user is exposed to the policy, and the quality of the policy. For exposure type, we compare our approach of showing critical states to two baselines: showing a one-minute rollout of the policy, and showing how the policy acts in random states, rather than critical ones. These two baselines are meant to approximate the states a user would happen to encounter as she observes and interacts with the robot over time.

For the quality of the policy, we have a policy π_A trained for 10,000 iterations, and another policy π_B that is trained for only 3,000 iterations. Both policies achieve similar performance on the task: π_A averages one crash per 700 timesteps, and π_B averages one crash per 640 timesteps.[4] But π_B fails in a few simple traffic scenarios, that π_A has learned to navigate successfully—including query states s_5 and s_7 (Fig. 5.5).

Fig. 5.5 shows a subset of the ten critical states per policy. Looking closely at the critical states of policy π_B (Fig. 5.5), we see the rightmost two states are false-positives, whereas all the critical states of policy π_A look reasonable. On average, the critical states of policy π_B are also of simpler driving scenarios, which suggests that it may not be able to handle more challenging ones.

Dependent Measures. We keep the same dependent measures as in the previous user study (Sec. 5.3.1), except we add two Likert questions that ask participants more specifically about how much they trust the policy with respect to critical states, and change the yes/no question for taking control to a five-point Likert question where higher means more likely to take control.

Hypothesis. Showing users the critical states of a policy establishes appropriate trust, compared to other approaches of exposing users to policies. Appropriate trust, in this setting, means that participants trust π_A over π_B, both in their Likert responses and in how often they choose to take control from the policy.

[3] We discretize this action space evenly into 200 possible actions, in order to compute critical states using the learned action-value function.

[4] Note that since the agent can only steer, and the other cars surrounding the agent are all driving slower than it, it will often encounter situations where crashes are inevitable.

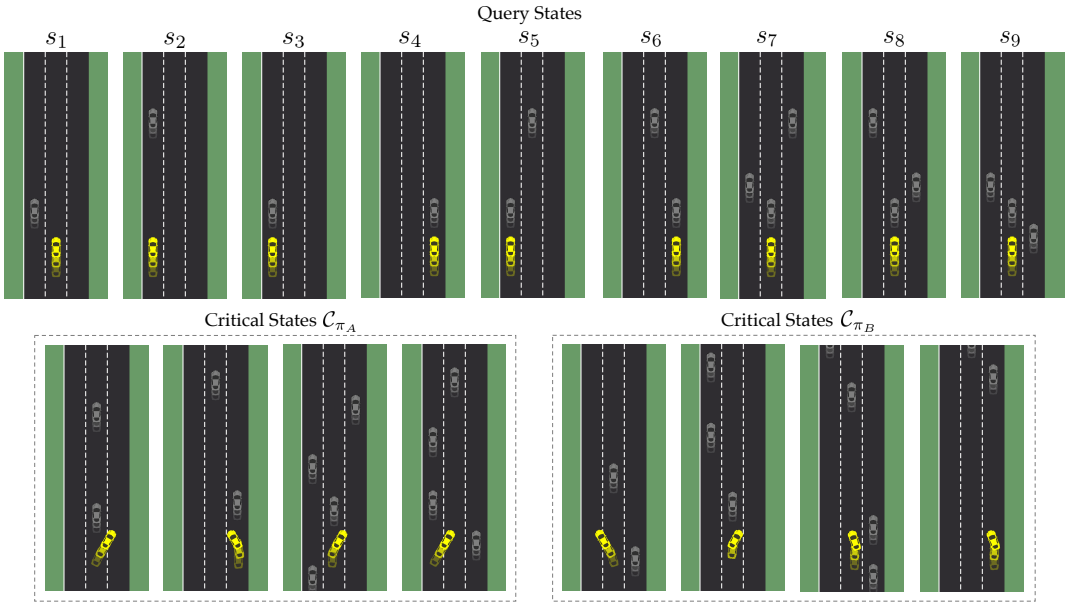


Figure 5.5: The query states and a subset of the ten critical states $\mathcal{C}_\pi$ shown in our main user study. The policy controls the steering of the yellow car. Query states s_1 and s_2 are not critical, but the rest are.

Subject Allocation. We used a between-subjects design for exposure type, and within-subjects for policy quality to reduce variance. We ran this experiment on a total of 60 participants across the three conditions, recruited via Amazon Mechanical Turk. The average age of the participants was 32.5 ($SD = 6.7$). The gender ratio was 0.27 female.

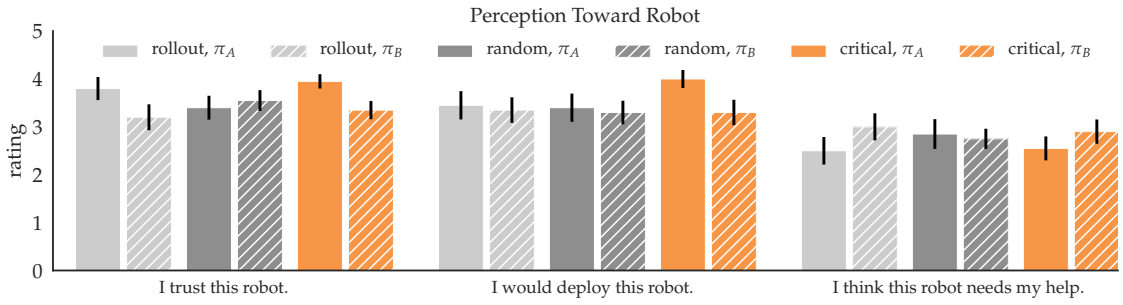


Figure 5.6: Ratings for Likert statements in Sec. 5.3.2, averaged across participants in each condition. Higher ratings mean higher agreement.

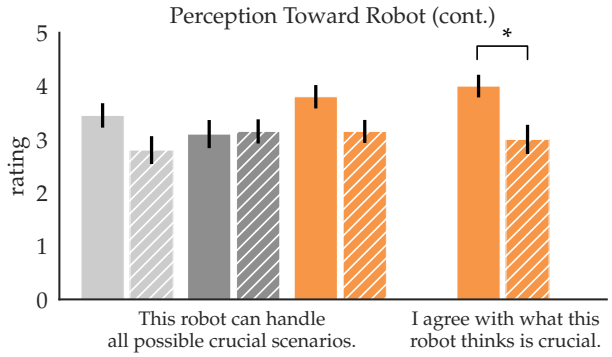


Figure 5.7: Additional ratings for Likert statements in Sec. 5.3.2.

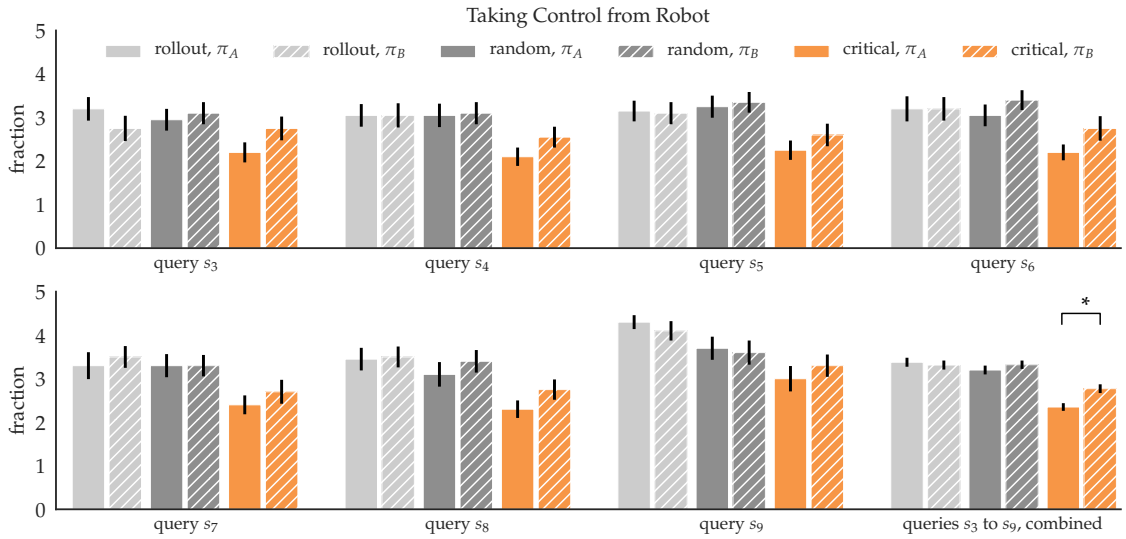


Figure 5.8: Participants' responses for whether they would take control of the policy at a particular query state (from Fig. 5.5). Results for s_1 and s_2 are omitted, since people overwhelmingly chose not to take control, regardless of which condition they were in.

Analysis

Subjective. We see that across all five questions, users who have seen the *critical states* of both policies tend to favor policy π_A, the better one (Fig. 5.6, Fig. 5.7). This trend is also visible for participants who see a one-minute rollout of each policy, but not as consistently. In contrast, when participants see how the policy acts in randomly-selected states, they rate policies π_A and π_B similarly, indicating that their trust is incorrectly calibrated.

We ran a two-way repeated-measures ANOVA, with exposure and policy quality as factors and user ID as a random effect, for each item except the question on agreement. We observe a weak interaction effect between exposure and policy quality for the question on trust ($F(2,57) = 2.37$, $p = 0.1$). We also ran a post-hoc Tukey HSD for each item, which confirmed the trend that participants in the critical-states condition favor the better policy, but this was not statistically significant.

We ran a one-way repeated-measures ANOVA, with policy quality as a factor and user ID as a random effect, for the question on agreement with critical states, and found a significant effect ($F(1,19) = 7.92$, $p = 0.01$).

Objective. We asked participants, for each of the query states (Fig. 5.5), whether they would take control from the policy at that state. Participants in the critical-states condition consistently choose to take control more in the case of the worse policy, π_B. We do not see this trend in either of the two baseline conditions (Fig. 5.8).

We also see that across all critical query states (s_3 through s_9), participants who saw the critical states of either policy are more likely to trust that policy and not take control of it, compared to participants who saw either a rollout of the policy or how it acts in randomly-selected states.

We ran a two-way repeated-measures ANOVA for the combination of participants' responses across all seven critical query states, and find a significant effect exposure ($F(2,57) = 5.57$, $p = 0.006$) and a significant interaction effect for exposure and policy quality ($F(2,777) = 5.30$, $p = 0.005$). We then ran a post-hoc Tukey HSD, which showed that when participants see the critical states of π_A and π_B, they take control significantly more for policy π_B ($p = 0.001$), but this is not true for either of the baseline conditions.

This suggests that by showing human end-users the critical states of a policy, we not only lead them to trust the policy more, but also enable them to appropriately calibrate their trust for good and not-as-good policies.

5.4 Discussion

Our user studies suggest that showing the critical states of a policy is a promising approach for not only building trust in the policy, but also for revealing whether it is trustworthy in the first place. This can be applied to any policy trained with a maximum-entropy-based approach.

The question is, what if a policy has incorrect critical states, but it performs very well, at least in the training environment. Should we trust this policy? Or should we not trust it, because the fact that it has incorrect critical states implies that it does not truly understand the task? This is an open question for future work. Our hunch

is that the latter is true—if a policy's critical states do not make sense, there are likely states (outside the training distribution) that it will not be able to generalize to.

The primary drawback of our approach is that it places significant responsibility and mental burden on the human end-user. For instance, we assume this end-user has domain knowledge about the task; this is likely true for supervising a self-driving car or robots in a factory, but might not be true for more complex tasks. In addition, identifying false-negative critical states requires the end-user to generalize correctly about what other states the robot considers as critical, given the ones they saw. One way to address this limitation is to reason about how humans do this generalization, and show the end-user how the robot acts in additional states (critical or not) to correct their understanding.

Nonetheless, this approach of showing critical states is a step toward giving human end-users a better chance of knowing whether or not to deploy a robot, and when to take control during deployment.

Part II

Transparency for Robot Learning

Chapter 6

Nonverbal Robot Feedback for Human Teachers

In Part I, we discussed approaches for efficiently improving humans' mental models of robots. Improved mental models not only helps human end-users interact with robots more safely and comfortably, but may also improve how well humans are able to *teach* robots. In this chapter, we explore how increased transparency of what a robot has learned, improves teaching from humans.

Robots can learn preferences from human demonstrations, but their success depends on how informative these demonstrations are. Being informative is unfortunately very challenging, because during teaching, people typically get no transparency into what the robot already knows or has learned so far. In contrast, human students naturally provide a wealth of nonverbal feedback that reveals their level of understanding and engagement. In this chapter, we study how a robot can similarly provide feedback that is minimally disruptive, yet gives human teachers a better mental model of the robot learner, and thus enables them to teach more effectively. Our idea is that at any point, the robot can indicate what it thinks the correct next action is, shedding light on its current estimate of the human's preferences. We analyze how useful this feedback is, both in theory and with two user studies—one with a virtual character that tests the feedback itself, and one with a PR2 robot that uses gaze as the feedback mechanism. We find that feedback can be useful for improving both the quality of teaching and teachers' understanding of the robot's capability.

6.1 Motivation and Background

If robots are to be useful to humans, they need to do more than optimize reward functions—they need to be able to figure out what reward functions should be optimized in the first place. Inverse Reinforcement Learning (IRL) [12] enables robots to infer preferences from human demonstrations. For instance, by collecting data of human drivers, we can infer a human-like driving style [27, 59].

Traditionally, IRL is applied in settings where data is collected offline from people who have no idea that a robot is supposed to learn from this data. When it comes to learning preferences for the purpose of assisting people—which is arguably the goal for most robots and AI agents—there is an opportunity to explicitly involve people in teaching the robot about what they want. It can be much more effective for me to actively teach my robot how to organize my kitchen, for instance, instead of having the robot collect data of me putting things away over and over again until it eventually figures out my preferences. When I teach, I get the chance to select examples that might be especially informative to the robot—ones that effectively illustrate the core of my approach.

Unfortunately, effective teaching is tricky even when we teach other people. We have to figure out what the person knows and does not know, what teaching strategy works best for them as an individual, etc. Teaching robots is monumentally harder. We have much poorer mental models of how robots learn compared to our mental models of how humans do. What is more, when we teach humans we receive a great deal of feedback from them. One traditional way we get feedback is through tests, by asking questions to probe the learner's understanding. More interestingly though, human students continually provide (nonverbal) feedback to the teacher *during the teaching process itself*. They look confused or bored, nod along, fidget, or gaze at various things [107, 108].

Our goal is for robots to provide similar feedback, at the same time as the teacher is providing examples. We take a step towards that in this paper based on a simple idea: to use the states that the teacher is in as opportunities to inform the teacher about what the robot expects them to do, according to the robot's current understanding of the task. We resort to *nonverbal cues* (e.g., gaze) as an intuitive and minimally-disruptive way for the robot to signal how it expects the teacher to act next. For example, imagine teaching the robot how to organize objects in a decluttering task (Fig. 6.1). Every time you pick up an object, the robot gazes at where it thinks the object should go. This helps you gain a better understanding of the robot's current hypothesis about your preferences, which helps you figure out a better next example to provide. It also helps you realize when the robot has finished learning, and you can stop teaching.

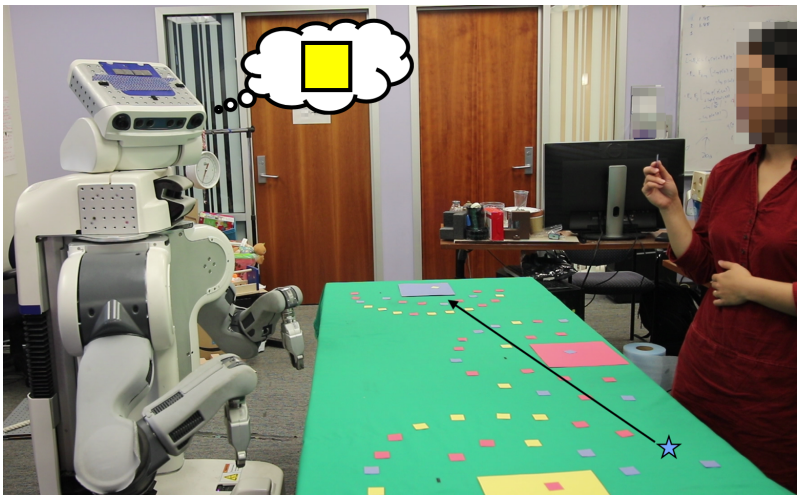


Figure 6.1: The robot uses its learned model to anticipate the teacher's next action (i.e., placing the object in the yellow bin) and then uses gaze to communicate its belief to the teacher.

Our main contributions are as follows: 1) we propose a form of feedback that robots can provide while teaching is ongoing, that consists of a prediction of the teacher's next action along with the confidence; 2) we provide a theoretical motivation for why this form of feedback should improve teaching effectiveness, by introducing an algorithmic teaching model that takes feedback into account; and 3) we test out this feedback with real people, both in an online user study with virtual learners and in-person with a real robot, where gaze is the feedback mechanism. We find that feedback can improve the teacher's estimate of the learner's understanding, quantifiably change the teacher's strategy, and result in higher learner accuracy. Results with real users support our theoretical analysis. With gaze, we find that these effects are stronger when participants know explicitly about the feedback they should expect—otherwise, some participants interpret gaze differently, e.g., as acknowledgement or as purely functional.

6.2 Related Work

Improving Quality of Human Input

Teaching robots via demonstration [109] is challenging; humans may have trouble providing useful demonstrations [51, 110] and knowing when to stop teaching [111]. Robots can take a more active role in learning, by asking for additional demonstrations

where they are uncertain [112–115] or know information is missing [116], or by asking clarification questions [117]. Robots can ask for different kinds of teacher input as well—e.g., labels, feature queries, or demonstrations—to maximize usefulness [35]. However, when robots are active learners, humans lose control over the teaching process, which can make them feel frustrated and disengaged [118, 119].

We take an orthogonal approach, in which humans maintain control of teaching, while the robot tries to be *transparent* about what it has learned. Transparency enables *algorithmic teaching*, in which the teacher's understanding of how the learner learns enables her to select teaching examples that optimally teach this learner.[1] Robot learners can be more transparent by demonstrating their current learned policy [120], allowing teachers to ask questions that probe their understanding [121, 122], or showing where they succeed and fail [111]. However, these approaches require people to stop teaching and separately test the robot. In contrast, we focus on how robots can provide feedback *at the same time* as humans are teaching—similar to the nonverbal feedback cues that human students give. This form of feedback requires minimal context switching from the teacher and does not add to the total interaction time. On the real robot, we implement this feedback as gaze.

Robot Gaze

Robots can use gaze to be transparent in social interactions with humans (see Admoni and Scassellati [123] for a detailed survey). Robot gaze can also be useful in human-robot collaboration tasks, by disambiguating referenced objects [124, 125], communicating which action the robot is about to take [126], and influencing the human's actions [127].

In this work, we also use robot gaze to communicate to humans, but specifically while the robot is *learning*. Thomaz and Breazeal [128] explored this for reinforcement learning (RL) agents: when agents use gaze to communicate what action they are about to take, humans are able to provide more informative rewards, thus speeding up learning. However, in RL the robot acts while the person observes. When robots instead learn from human demonstrations, the opposite is true, which makes the robot's learning opaque. Taking inspiration from the RL setting, we aim at having the robot always convey what it thinks is the optimal action to take, so that the person can adjust their demonstrations appropriately.

[1] Since the teacher knows the task, algorithmic teaching is typically more efficient than active learning [33].

6.3 Approach: Nonverbal Feedback

6.3.1 Assumptions on Robot Learning Algorithm

We focus on robot learners that infer *reward functions* from demonstrations via IRL [12]. The benefit of IRL is that this underlying reward function typically generalizes better across tasks, compared to directly learning a mapping from states to actions, as in behavior cloning [129,130]. In addition, humans are naturally inclined to infer the objectives of other agents [16, 19], and thus might expect robot learners to do the same. We parameterize the reward function $\mathcal{R}_\theta$ as a linear combination of reward features $\phi(\cdot)$ with weights θ,

$$\mathcal{R}_\theta(s,a) = \theta^\top \phi(s,a)\,,$$

where s is the state and a is the action. There is no limitation on what these reward features can be, so this assumption is not restrictive [59].

The robot maintains a belief $b(\theta)$ over reward function parameters. We model humans as providing demonstrations (s,a) that are approximately optimal, according to a Boltzmann distribution [131]. This induces the following observation model that links human actions to the reward parameters,

$$p(a|s,\theta) \propto e^{\beta Q^*_\theta(s,a)}\,, \tag{6.1}$$

where β specifies the level of suboptimality and $Q_\theta(s,a)$ is the action-value function: the discounted sum of future rewards, after taking action a in state s and acting optimally thereafter with respect to $\mathcal{R}_\theta$. As in Bayesian IRL [56], the robot uses this observation model to update its belief over θ:

$$b'(\theta) \propto p(a|s,\theta)\; b(\theta)\,. \tag{6.2}$$

6.3.2 Generating Feedback

We propose and investigate a form of feedback where at every step, the robot communicates what it believes the optimal action is. Intuitively, this should help human teachers understand what the robot knows and does not know as they proceed through teaching the task, enabling them to adapt what they teach and recognize when they can stop teaching. In addition, this form of feedback is minimally disruptive and allows teachers to maintain control of teaching, in contrast to tests of comprehension and active learning, respectively. We leave open the question of how this feedback should be communicated; we experiment with movement of a virtual avatar and gaze on a real robot.

Feedback Target

The robot first uses its current belief over reward parameters to predict the human's next action,

$$\hat{a} = \arg\max_{a} \int_{\theta} p(a|s,\theta)\, b(\theta)\, d\theta\,, \tag{6.3}$$

with the observation model from Eqn. (6.1). Based on this prediction, it determines what the most likely next state is, i.e., where the human will go next:

$$\hat{s} = \arg\max_{s'} p(s'|s,\hat{a})\,. \tag{6.4}$$

In a deterministic environment, this is $\hat{s} = f(s,\hat{a})$, with f being the dynamics. The robot then communicates this prediction to the human teacher, for instance by gazing in the direction of $\hat{s}$.

Feedback Speed

If the robot can control the *speed* of feedback (e.g., the speed at which it moves its hand or gazes), it can use this to convey how *confident* it is in its prediction:

$$v = v_{max}\; p(\hat{s}|s,\hat{a}) \int_{\theta} p(\hat{a}|s,\theta)\, b(\theta)\, d\theta\,, \tag{6.5}$$

with v_{max} being the maximum allowed speed.

Grounding in Our Experimental Domain

To make this more concrete, consider the domain of decluttering: objects need to be sorted appropriately into bins, and only the human knows the correct sorting mechanism. States are locations of objects and bins, and actions place objects into bins. Every action a corresponds to a particular bin B_a, and we assume the environment is deterministic; taking action a in state s means putting the object from location s into B_a. $\phi(s,a)$ is a feature vector that consists of descriptors of object-bin match, e.g., distance, color match, shape match, etc.

Each human action teaches the robot about the relationship between the features and the reward, i.e., about the correct weights θ^*. Once the human has selected an object, the robot predicts, according to its current belief, which action $\hat{a}$ the human will take next, and indicates $B_{\hat{a}}$, for instance via gaze. Each θ assigns different probabilities to different bins, and the robot combines all this information to compute a confidence in its estimate of bin $B_{\hat{a}}$, which it uses to adjust its feedback speed.

6.4 Analysis in Theory: Why Will Feedback Help?

6.4.1 Model of Human Teachers That Incorporates Feedback

To provide a theoretical justification for feedback, we construct potential models of human teaching that incorporate feedback, and analyze how feedback improves teachers' decisions. These models are based on algorithmic teaching [29, 30, 33], and thus have two components: tracking the learner's state (e.g., what does the learner know), and selecting informative demonstrations based on this.

Tracking Learner State

We assume our teacher knows how the learner performs an update after every example via Eqn. (6.2). However, the teacher is still missing information about the learner, in particular the learner's *prior* belief b_0, and the feature space Θ that the learner assumes—which may differ from the teacher's. A sophisticated teacher is aware of this *uncertainty*. At every step, she has a belief over what the robot's belief might be, that takes into account the robot's learning updates, its feedback, and the uncertainty over which prior and feature space the robot is using.

After the first time step, this teacher computes the probability of the robot's new belief given the robot's feedback (x_0, v_0) for the object she picked up, s_0, and the action she took, a_0:

$$\begin{aligned} &p(b_1|s_0, a_0, x_0, v_0) \\ &= \int_{b_0,\Theta} \underbrace{p(\Theta)p(b_0|\Theta)}_{\text{teacher priors}} \underbrace{p(x_0, v_0|b_0, s_0)}_{\substack{\text{feedback generation} \\ \text{Eqn. (6.7) (6.8)}}} \underbrace{p(b_1|b_0, s_0, a_0, \Theta)}_{\substack{\text{robot learning} \\ \text{Eqn. (6.2)}}} d\Theta db_0 \end{aligned} \tag{6.6}$$

Since we assume the teacher knows how the learner performs belief updates, for the robot learning distribution, $p(b_1|b_0, s_0, a_0, \Theta)$, the teacher places all probability mass on the correct new belief according to Eqn. (6.2). For the feedback generation distribution, $p(x_0, v_0|b_0, s_0)$, we use how the robot actually generates feedback. From x_0, the teacher recovers $\hat{a}_0 = f^{-1}(s_0, x_0)$, the action that the robot predicted, based on inverse dynamics f^{-1}. From there, the teacher computes the probability of that action under the belief b_0:

$$p(x_0|b_0, s_0) = p(\hat{a}_0|b_0, s_0) = \int_\theta p(\hat{a}_0|s_0, \theta)\, b_0(\theta)\, d\theta\,. \tag{6.7}$$

For v_0, the teacher assigns probability based on a Gaussian around her predicted speed for $\hat{a}_0$:

$$p(v_0|b_0, s_0, x_0) = p(v_0|b_0, s_0, \hat{a}_0) = \mathcal{N}(v_0|v_{max}\; p(\hat{a}_0|b_0, s_0), v_{var})\,. \tag{6.8}$$

The teacher keeps incorporating such updates based on robot feedback for each new demonstration. In practice, we approximate this update by initializing the teacher's priors $p(\Theta)$ and $p(b_0|\Theta)$ with sampled learner prior beliefs b_0 and possible feature spaces Θ, and iteratively updating this set of samples with the evidence, analogous to running a particle filter [132] (see Sec. 6.4.2 for details).

We also consider a less sophisticated teacher model, who does not account for the uncertainty over the robot's prior and feature space. Rather than maintaining a belief over beliefs, this *iterative* teacher starts with a uniform belief over what the robot's reward estimate might be, and updates it at every step. First, the teacher updates based on the feedback:

$$b'_1(\theta|s_0, x_0, v_0) \propto p(x_0, v_0|\theta, s_0)\, b_0(\theta), \tag{6.9}$$

with the feedback probabilities computed as in Eqn. (6.7) and Eqn. (6.8), but for a single θ. Next, the teacher shows a_0 and accounts for the fact that the robot will learn:

$$b_1(\theta|s_0, a_0, x_0, v_0) \propto p(a_0|\theta, s_0)\, b'_1(\theta). \tag{6.10}$$

Motivated by the "win stay, lose shift" strategy in cognitive psychology [133], iterative teachers only update if the feedback disagrees with their current belief in the learner's state. In other words, if the learner predicts the bin that the teacher's maximum a posteriori (MAP) estimate of the learner's θ deems most likely, and the corresponding velocity for this θ_{MAP} is within $0.05 * v_{\max}$ of the learner's feedback speed, then the teacher does not update based on this feedback. Thus, if the teacher has a perfect model of how the learner learns, then whether the learner provides feedback or not, the teacher would maintain the same belief over the learner's hypothesis, and thus teach in the exact same way. If we did not make integrating feedback conditional, then this teacher would potentially teach *worse* in this situation, since the additional feedback-based updates cause the teacher's estimate to diverge from the true one.

Selecting Examples Based on Current Learner State

At every step, our teacher uses the tracked learner state to select the most informative example to give next. This is the example that leads to the largest increase in learner performance. The iterative teacher teaches:

$$(s_t^*, a_t^*) = \arg\max_{s_t, a_t} g(b_{t+1}(\theta|s_t, a_t)), \tag{6.11}$$

where $g(\cdot)$ computes the learner's expected performance on the task (if its belief were b_{t+1}) and $b_{t+1}(\theta|s_t, a_t) \propto p(a_t|\theta, s_t)\, b_t(\theta)$, according to the learner's learning update via Eqn. (6.2).

The uncertainty-aware teacher does the same, but in expectation over the robot's current belief and the feature space the robot uses:

$$(s_t^*, a_t^*) = \arg\max_{s_t, a_t} \mathbb{E}_{b_t, \Theta} \left[g(b_{t+1}(\theta | s_t, a_t)) \right] \tag{6.12}$$

6.4.2 Impact of Feedback

We now investigate how much feedback helps for our models of human teaching. If the teacher has a perfect model of the learner, then feedback is not necessary [30]. So, we focus on the kinds of mismatches that might occur—the learner might start off with a bad prior, or not know the correct feature space. For these simulated experiments, we randomly sampled N bins and O objects; each bin and object is represented by a d-dimensional feature vector. We set $N = 3$, $O = 50$, and $d = 3$.

Manipulated Variables. We manipulate the teacher type: *iterative*, *uncertainty*-aware, and *random*. The *random* teacher is a baseline, that chooses random teaching examples rather than maximally informative ones. The learners vary along two factors: whether they provide feedback or not (*feedback* versus *no-feedback*), and in terms of whether the teacher-learner mismatch is *none*, the prior (*prior mismatch*), the learner missing a feature (*feature mismatch*), or the learner learning a separate θ for each bin (*reward generalization mismatch*, i.e., reward features are computed in terms of only the object s, as in $\phi(s)$).

Learner Models. In the *prior mismatch* condition, we bias the learner toward a randomly-selected θ' by setting the prior as the softmax of the distance between θs:

$$p(\theta) \propto e^{\beta' \theta^\top \theta'}, \tag{6.13}$$

with $\beta' = 50$ for a fairly strong bias. In the *feature mismatch* condition, the learner ignores the last feature dimension. In other words, the set of θs that the learner considers all have $\theta_d = 0$. In the *reward generalization mismatch* condition, the learner learns a separate θ for each bin. So the space of θs the learner is considering now has dimensions $B * d$.

For learning updates, we set $\beta = 20$ for a learner that assumes demonstrations are close to optimal. For feedback, we set $v_{\max} = 1.0$ and enforce a minimum speed of 0.05. We sampled 1000 θs to approximate the learner's belief.

Teacher Models. In the *prior mismatch* condition, the uncertainty-aware teacher considers $P+1$ possible learner models, one of which is the default (i.e., uniform prior b_0); the other P are biased towards different θs, including the one that the learner is actually biased towards. In the *feature mismatch* condition, the uncertainty-aware teacher considers $d+1$ possible learner models: one with all features, and d that are each missing a different one of the d features. Finally, in the *reward generalization*

mismatch condition, the uncertainty-aware teacher considers two possible learner models, one of which has reward features $\phi(s, a)$ that depend on both the object and bin, and the other of which has reward features $\phi(s)$ that depend only on the object. We assume uncertainty-aware teachers start off believing that the learner likely has a uniform prior b_0, considers all d features, and uses reward features $\phi(s, a)$—so we set the teachers' priors (i.e., $p(b_0|\Theta)$ and $p(\Theta)$) to place equal probability on all of the models, except ten times more probability on this most-likely learner model. For teacher updates, we assume $v_{\text{var}} = 0.0025$.

Learner Performance. We measure learning performance with expected soft classification error,

$$g(b_t(\theta)) = \mathbb{E}_{\theta \sim b_t(\theta)} \left[\frac{1}{|S|} \sum_{s \in S} p(a^*(s)|s, \theta) \right], \tag{6.14}$$

where $a^*(s) = \text{argmax}_a \mathcal{R}_{\theta^*}(s, a)$ denotes the correct bin to place the object in.

Analysis. As one might expect, we found that feedback has different effects, depending the teacher-learner mismatch and whether the teacher does iterative or uncertainty-aware updates. We found two main positive effects: feedback allows the teacher to 1) select more effective teaching examples, and 2) track the capabilities of the learner more accurately.

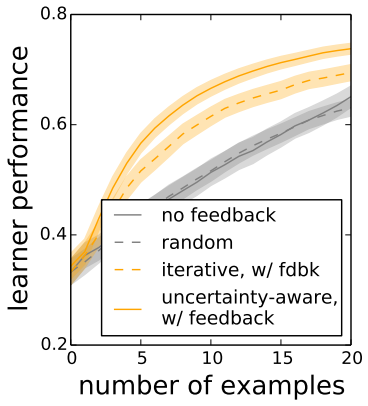


Figure 6.2: For mismatched priors, algorithmic teaching with feedback achieves higher learner performance (6.14). Standard error bars are across 100 trials.

Feedback leads to more effective teaching for only the *prior mismatch* condition (Fig. 6.2). Uncertainty-aware teachers very quickly narrow down which learner model is correct, and are able to nearly-perfectly estimate the learner's performance (Fig. 6.3). This makes the potentially strong assumption that the uncertainty teacher's possible learner models contain the true one. This is reasonable though, in the case of *feature mismatch* and interpretable reward features, because then the teacher could just be reasoning over the power set of features.

In the *features mismatch* and *reward generalization mismatch* conditions, the learner cannot actually learn the task because it is reasoning over the incorrect space of θs, but there is still a benefit to the teacher estimating the learner's performance correctly: they have a better idea of when to stop teaching (i.e., when their estimated performance stops increasing), and they have a more accurate estimate of the learner's performance at

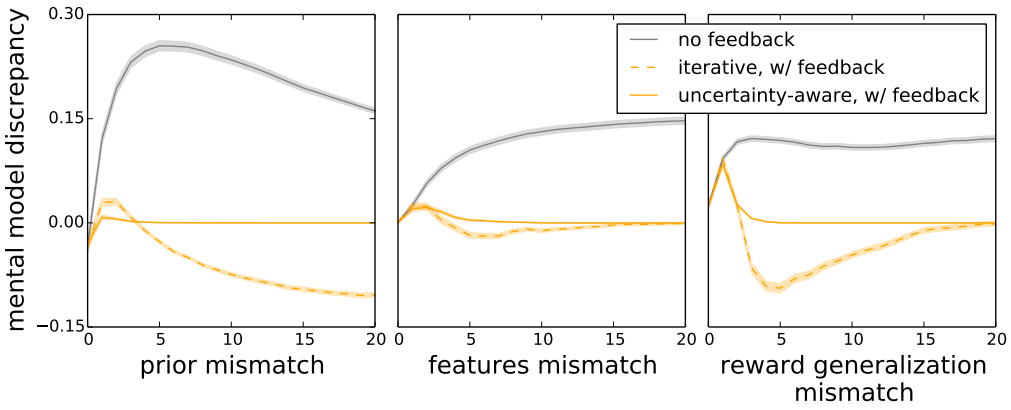


Figure 6.3: With feedback, iterative and uncertainty-aware teachers (orange) can track the learner's state accurately for most mismatch conditions. In contrast, teachers significantly overestimate learner performance when there is no feedback. Standard error bars are computed from 100 trials with ten random teaching sequences each. The y-axis is the teacher's estimate of learner performance minus true performance (6.14); the x-axis is the number of teaching examples.

the end of training. They also have a better chance of identifying which feature(s) the learner is missing, which gives them a better idea of the learner's capabilities for future tasks.

Summary. Our analysis on algorithmic-teaching-based human models suggests that feedback helps teachers. When the learner can learn the task, feedback makes it easier for teachers to select effective teaching examples. Feedback also improves the teacher's estimate of the learner's capabilities, so when the learner cannot learn the task, feedback enables users to realize this.

6.5 Analysis in Practice: Feedback Helps

We next investigate whether real human teachers also benefit from feedback. In this section, we test the benefits of the feedback itself with virtual learners that explicitly predict the teacher's action with varying confidence. In the next section, we test whether these benefits still exist when the feedback is realized through *gaze* on a real robot.

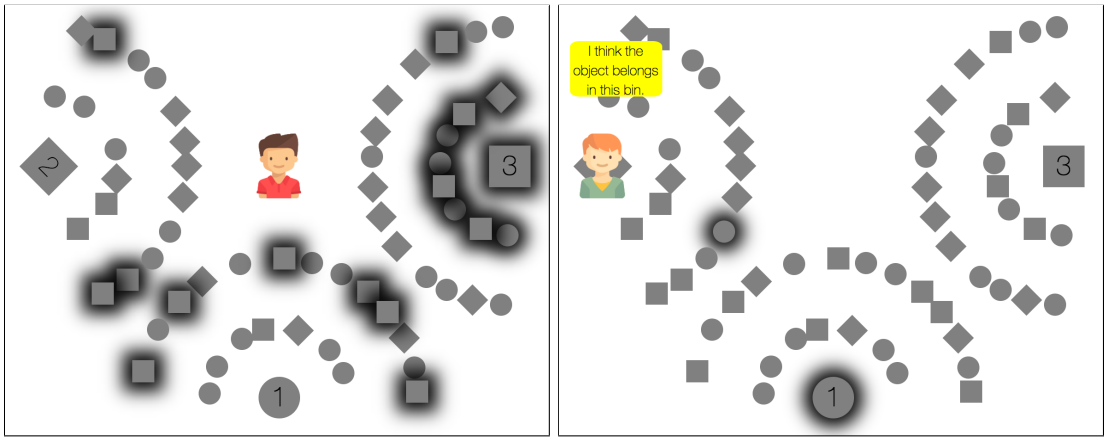


Figure 6.4: Amazon Mechanical Turk teaching interface, where correct object-bin pairings are visualized (left, bin 3), and learners' feedback explicitly signals their prediction (right).

6.5.1 Design

We recruited 87 participants (ages 22-71, 41% female) on Amazon Mechanical Turk (AMT) to teach a decluttering task to an agent. Participants demonstrated correct object-bin pairings to the agent; we instructed them to give as few teaching examples as possible while still ensuring that the agent learns the correct sorting rules. The decluttering setup (Fig. 6.4) consisted of three numbered bins surrounded by two rings of objects, with the following sorting rules: 1) an object lying on an inner ring belongs to the closest bin, and 2) an object lying on an outer ring belongs to the bin with the same shape as it.

Users clicked on the object they wanted to demonstrate, and it would be moved into the correct bin. When feedback was activated, the learner avatar would move toward its best guess for the corresponding bin and declare that it thought the object belonged there (Fig. 6.4, bottom). Then the object would be moved to its bin, and the learner would acknowledge whether they were mistaken.

Manipulated Variables. We manipulated the *feedback*: no feedback from the learning agent, full feedback that indicates the learner's best-guess bin with variable speed corresponding to its confidence level, or partial feedback to indicate the best-guess bin with a fixed speed. We also manipulated *learner prior belief*: a uniform prior over all weights in the teacher's feature space Θ, a biased prior over Θ, or a uniform prior over weights in a mismatched feature space different from the teacher's. We did not conduct the 3 by 3 factorial—instead, we analyze the impact of prior condition separately from the impact of partial feedback, since we did not hypothesize

any interactions there. We thus did a 2 by 3 study with full feedback and no feedback, and only tested the partial feedback on the prior belief condition.

Implementation Details. We generated 1024 θs for the learner to reason over, including those that corresponded to conceptually intuitive sorting rules (e.g., objects belong to bins with their same shape). In the uniform prior case, the belief was uniform over these θs, and in the biased prior case, the learner heavily preferred sorting all objects into their closest bin. In the mismatched features condition, we removed the shape dimension feature in the learner's belief space.

Dependent Measures.

Objective. We measure the *learner performance* (6.14) throughout teaching. We record the *teaching sequence length* and measure proxy metrics for teaching strategy, e.g., the *proportion of objects demonstrated from each ring.*

Subjective. We also ask open-ended and Likert scale questions about the confidence of the teacher in the learner's understanding, their ability to track the learner's progress, the helpfulness of feedback, and the effects of feedback on teaching.

Combined. In addition, we compute a *mental model discrepancy* metric [119]: the human's Likert estimate of the robot's understanding of a sorting rule (scaled to the 0-1 range), minus the learner performance for objects classified by that rule.

Hypotheses. We hypothesize that for all learner prior conditions, **H1**: Feedback will allow the teacher to better track the learner's progress and understanding, **H2**: The teacher will adapt their strategy according to their improved estimate of learner capabilities, and **H3**: This adjustment, enabled by feedback, will ultimately result in increased learning performance.

Subject Allocation. Participants were randomly allocated between-subjects across the learner prior conditions. Full versus no feedback was within-subjects, to ensure a direct comparison, since people may have significantly different teaching strategies and capabilities. With no feedback, teachers had to make up a plan and execute open-loop. With feedback, they could adjust this plan based on what they found out about the learner via feedback. Because of this, we put the open-loop plan first—if users were able to see the feedback first, they would keep the robot's limitations in mind even for the open-loop case, which could bias the results. To ensure understanding of both the interface and the sorting rules, participants completed a practice task different from the main teaching task, and had to pass a quiz on the sorting rules before teaching.

6.5.2 Results

H1: Feedback improves tracking of robot understanding. In learner prior conditions with high mental model discrepancy at the end of teaching without

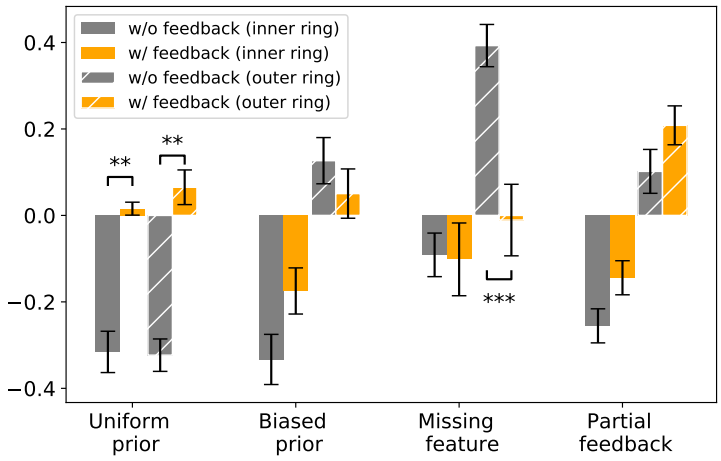


Figure 6.5: Discrepancy in teacher's belief of actual (final) learner ability to sort inner and outer ring objects is lower in the presence of feedback. Lower magnitude is better; positive values mean that the teacher overestimates the learner capability, and negative values mean they underestimate. (**) indicates $p < 0.005$ and (***) $p < 0.0005$.

feedback, adding feedback reduces this discrepancy (Fig. 6.5). A 2 by 3 factorial repeated-measures ANOVA with feedback and prior belief as factors showed a significant interaction effect on both outer ($F(2, 84) = 19.78, p < .0001$) and inner ($F(2, 84) = 4.51, p = 0.013$)) ring mental model discrepancy. The post-hoc Tukey HSD found that feedback significantly decreases discrepancy in the uniform prior condition for both the outer ($p = 0.001$) and inner ($p = 0.002$) ring rules, and in the missing feature condition for outer ($p < 0.0001$). Subjective Likert responses also support this hypothesis; see the analysis at the end of this section for details.

H2: Feedback changes teaching strategy. In the biased prior and the missing feature conditions, feedback leads teachers to demonstrate more outer-ring objects (Fig. 6.9); an ANOVA found a significant main effect for feedback on this measure ($F(1, 85) = 31.49, p < .0001$). This is reasonable: in both conditions, the learner has trouble learning the outer ring rule, participants give more teaching examples: an ANOVA on the teaching sequence length found an interaction effect between the feedback and learner mismatch ($F(2, 63) = 6.613, p = .0025$). A post-hoc Tukey HSD found that the number of teaching examples is significantly higher for the missing feature case when people receive feedback—they persist in teaching the learner the outer ring rule, until they finally realize it is impossible.

H3: Feedback improves learning performance. Feedback during teaching

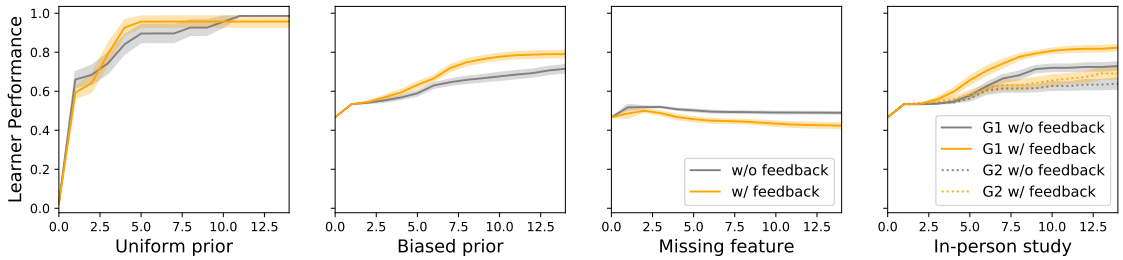


Figure 6.6: Learner performance (6.14) versus teaching iteration in the online study (left three) and in the in-person study with a biased prior (right-most). The x-axis is the number of teaching examples.

leads to improved learner performance in the biased prior case (Fig. 6.6), in line with our results on ideal teacher models (Sec. 6.4). We ran a 2 by 3 factorial repeated-measures ANOVA on final learner performance with feedback and prior belief as factors, and number of examples as a covariate. We found a significant interaction effect between feedback and prior belief ($F(2, 1925) = 29.74$, $p < .0001$). A post-hoc Tukey HSD found that feedback significantly improves performance for the biased prior case, significantly decreases it for missing feature, and makes no difference for uniform prior. Since the learner cannot learn the task at all in the missing features condition, it is more important that the teacher recognizes the robot's learning limitations—which feedback does help with.

We found no significant differences between full versus partial feedback in terms how effective teachers were (i.e., final learning performance), the proportion of demonstrated objects from the outer ring, and the number of teaching examples shown. However, learners with partial feedback do tend to overestimate learner performance for the outer ring rule (Fig. 6.5), possibly because they assume the learner is consistently confident in its prediction.

Analysis of Likert Responses

We ran a two-way repeated measures ANOVA on each of the subjective measures (i.e., Likert questions) with the feedback and prior belief as factors. The full details are reported in Table 6.1. For clarity, shorthand for the questions will be used (full questions can be found in Table 6.1.

Tracking robot understanding. In the uniform prior condition, when the learner is able to learn both the inner and outer ring sorting rules, participants' subjective Likert responses indicate that when they were given feedback, they were significantly more confident that the learner learned correctly (Fig. 6.7, *inner* and *outer*). A

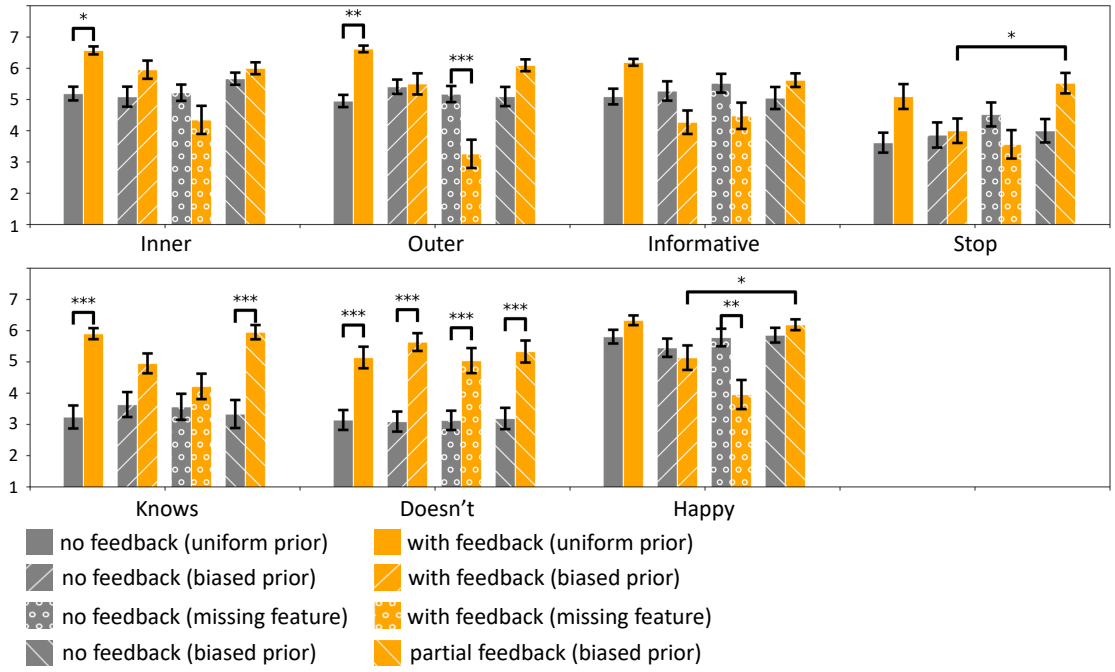


Figure 6.7: Likert scale responses from the AMT study, in which users taught learners with and without feedback. (*) indicates $p < 0.05$, (**) indicates $p < 0.005$ and (***) indicates $p < 0.0005$. The full Likert statements can be found in Table 6.1.

post-hoc Tukey HSD found a significant difference between feedback and no feedback for both *inner* ($p = 0.004$) and *outer* ($p = 0.004$).

Similarly, in the missing feature condition, when the learner is not able to learn the outer ring rule, participants who received feedback were more aware of this; a post-hoc Tukey HSD found a significant difference between feedback and no feedback for *outer* ($p = 0.0003$).

Across all prior belief conditions, participants found it overwhelmingly easier to recognize what the learner *does not* understand; the ANOVA found a significant main effect for feedback for *doesn't*. Participants also generally found it easier to recognize what a learner understands; a post-hoc Tukey HSD found a significant difference between feedback and no feedback for the uniform prior ($p < 0.0001$) and partial feedback ($p < 0.0001$) conditions.

User satisfaction. For *happy*, a post-hoc Tukey HSD found a significant difference between feedback and no feedback for the missing feature condition ($p = 0.0011$). This makes sense, because when feedback was given in the missing feature condition, participants were clearly aware that the robot was unable to learn the task, despite

Table 6.1: ANOVA Results for AMT Study Likert Questions.

Statement	Effect	F-score	p-value
Inner: "[Learner] correctly learned the rule that if an object lies in an inner ring, it belongs to the closest bin"	interaction	$F(3, 83) = 7.99$	$p < 0.0001$
Outer: "[Learner] correctly learned the rule that if an object lies in an outer ring, it belongs to the bin with the same shape"	interaction	$F(3, 83) = 14.12$	$p < 0.0001$
Informative: "I found it easy to choose informative teaching examples for [Learner]"	none	N/A	N/A
Stop: "It was easy to know when to stop teaching [Learner]"	none	N/A	N/A
Knows: "It was easy to tell what [Learner] knows about the task rules"	interaction	$F(3, 83) = 3.89$	$p = 0.01$
Doesn't: "It was easy to tell what [Learner] *doesn't* know about the task rules"	interaction	$F(1, 83) = 87.7$	$p < 0.0001$
Happy: "I would be happy to teach [Learner] again"	interaction	$F(3, 83) = 6.14$	$p = 0.0008$

their efforts to teach it.

Full versus partial feedback. An ANOVA on each of the subjective measures with the feedback level as a factor (full feedback versus partial feedback) found significant differences between the partial and full feedback condition for only the *stop* ($F(1, 47) = 8.43, p = 0.0059$) and *happy* ($F(1, 47) = 5.57, p = 0.02$) questions. The average for the partial feedback condition was higher in both cases. We hypothesize that this is because participants found the partial feedback (with constant speed) to be easier to reason about. This suggests that there is a need for investigating other mechanisms for expressing feedback confidence.

Summary

Overall, our results with human teachers were remarkably similar to what we saw in simulation with our teacher model: 1) feedback helped in the biased prior condition; and 2) although feedback did not help in the uniform prior or missing feature conditions, it helped reduce discrepancy between the teacher's model of the learner and the actual learner.

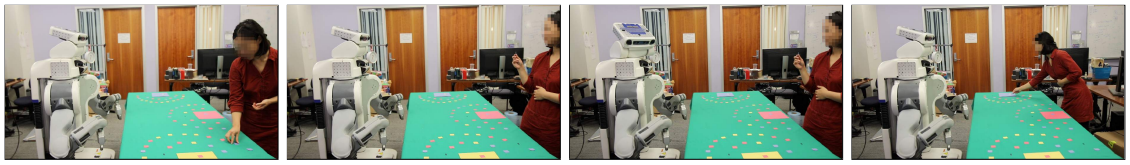


Figure 6.8: Real-world experiment flow (from left to right): the participant 1) selects the object to demonstrate, 2) picks it up and shows it to the PR2, 3) observes the PR2's gaze feedback, and 4) places the object in the correct bin.

6.6 Analysis in Practice: Robot Gaze

In the real world, gaze cues are more natural and less disruptive than explicit communication. We conducted an in-person study with a PR2 robot, to investigate whether more subtle gaze feedback cues are indeed interpretable and usable by humans. Thus, participants were not told beforehand of the purpose of the robot's gaze patterns.

6.6.1 Design

We replicated the virtual interface in a real-world decluttering task, using the same object-bin layout. Instead of a virtual learner that explicitly moves to its best-guess bin, in this study a PR2 robot relies only on gaze for feedback, changing its head orientation at varying speeds (Fig. 6.8).

We chose to only test the biased prior condition, because this is where feedback has the largest positive impact on learner performance—for both ideal teacher models and human teachers of virtual learners that provide feedback explicitly. We maintain the same hypotheses from the AMT study.

We recruited 17 users (ages 18-26, 47% female) from the general student population of our university—a limitation we discuss in Sec. 6.7. Each taught the PR2 the same task twice: first with no feedback, and then with gaze feedback. Participants were told that the robot was reset between the two tasks, so they would not assume the agent retained knowledge from previous demonstrations.

6.6.2 Results

Nine participants realized the gaze was related to the robot's belief of the sorting rule. The others misunderstood the robot's gaze as either observing where the demonstrated object was selected from or an acknowledgement of its understanding (since its gaze at the bin in front resembles a nod).

We thus report findings separately for those who understood the true intent of the gaze (G1) and those who did not (G2). For G1, feedback indeed improves learner performance (Fig. 6.6). A repeated-measures ANOVA confirmed this ($F(1,8) = 9.53$, $p = .01$). Feedback also increases the proportion of examples from the outer ring, as in the AMT study (Fig. 6.9). A repeated-measures ANOVA on the proportion of outer ring objects found a significant main effect for feedback ($F(1,8) = 9.25, p < .02$). None of this is true for G2. Moreover, virtually no participants were aware of the variations in speed of the robot's head motion, which suggests that more natural gaze patterns (e.g., modulating acceleration) or more noticeable gaze pattern differences should be explored.

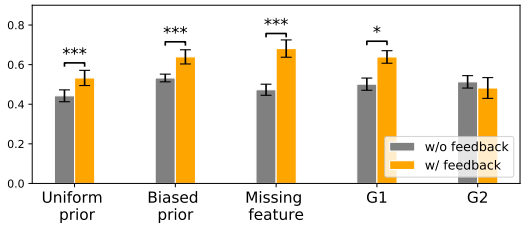


Figure 6.9: Proportion of demonstrated objects from the outer ring on AMT (left) and in-person (right). (*) indicates $p < 0.05$ and (***) $p < 0.0005$.

6.7 Discussion

Summary. We find that our proposed form of nonverbal robot feedback, predicting the teacher's next action, helps improve the teacher's effectiveness and mental model of the learner. Findings in practice echo those from our algorithmic teaching model. We also find communicating confidence does not help significantly in practice, and about half of users do not naturally interpret gaze as the intended form of feedback.
Limitations and Future Work. Our work is limited in several ways. The main limitation is that our experiments are on a relatively simple, non-sequential task. In the real world, for more complex tasks with larger action spaces, gaze may have less communicative power; investigating effective combinations of gaze and other feedback channels is a promising future direction. In addition, due to the small and biased sample, the results of our study with gaze on the PR2 should be interpreted as trends. Further work is necessary to explore the viability of gaze as a feedback mechanism, e.g., explicitly explaining to users the purpose of gaze, or making the gaze itself more human-like.

Finally, the form of feedback we study only improves learning when the learner can learn the task, but the teacher's model of the learner is not perfect (e.g., a mismatched prior). However, even when the task is not learnable, this form of feedback is still important in helping teachers recognize the robot's limitations and correctly estimate its (lack of) learning.

Chapter 7

Conclusion

It is important for human end-users to have accurate mental models of the robots they interact with, to ensure safe and seamless interactions. Unfortunately, this process typically takes a while, since it depends on which situations the human gets to interact with or observe the robot in, and this is left up to chance.

The research in Part I shows it is possible to speed up how long it takes end-users to form accurate mental models. Our key insight is that a robot's actions influence not just the physical world, but also what a person thinks about it. Thus, robots have the opportunity to give *informative* examples of behavior that are especially useful for improving end-users' mental models.

In Chapter 2, we introduced a framework for selecting informative examples, by framing the problem as an MDP over the human's belief, in which the robot is rewarded for giving examples of behavior that bring the human's belief closer to the true robot model. The human's belief could concern a robot's objectives, the dynamics constraints it is subject to, and/or its policy. Each of these assists with different goals of transparency. Knowing a robot's objectives helps users anticipate what this robot will do, even in novel scenarios (Chapter 3); additional understanding of a robot's dynamics constraints enables users to understand why it failed (Chapter 4); and understanding a robot's policy enables users to decide in which situations to trust the robot or not (Chapter 5).

Accurate mental models of robots not only make human-robot interaction more safe and seamless, but also help humans teach robots more effectively. Part II found that even a simple form of feedback from the robot, predicting the human teacher's next action, helps human teachers both better estimate the robot's capabilities and select more effective examples for teaching it (Chapter 6).

Future Directions

A main challenge of this work is that we do not know the true transition function of the MDP, i.e., how humans update their belief about the robot given observations of its behavior. In practice, the choice of the human model $\mathcal{H}$ can have a significant impact on the informativeness of examples for actual humans (Chapter 3). Thus, a promising direction for future work is to make the communication process more interactive—while the robot shows examples of its behavior, it should also be estimating what the human's current mental model of it is, so that it can use this to refine which future examples of behavior it shows. This is analogous to Chapter 6, in which we showed that when (human) teachers do not have a perfect model of how the (robot) learner learns, feedback from the learner helps teachers select more effective teaching examples.

A robot has the most flexibility for showing informative examples when it not only has control over its own trajectory, but also over the layout of its environment. Allowing for the latter typically requires showing these examples via a simulator, since it is often impossible for a robot to arrange its environment in the physical world (e.g., setting a particular configuration of nearby cars, pedestrians, and traffic signals). Then the realism of the simulator directly impacts the accuracy of the mental models that people form. It would be worthwhile to study how well people are able to apply the mental model they form of the robot in simulation, to interactions with the real robot. Effective transfer may require using more realistic simulators than, for example, the simple top-down driving simulators that we used for our experiments in Chapter 3 and Chapter 4.

Finally, the works in this thesis all use motion as the robot's communication channel, since humans naturally draw conclusions about robots based on their motions, and motions are more precise. A natural extension is to combine this with other communication modalities, such as speech and visualization, that have complementary strengths compared to motion—speech is not as precise but is more concise, and visualizations are longer and less natural to process but can be more rich.

Bibliography

[1] C. Heyer, "Human-robot interaction and future industrial robotics applications," in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2010.

[2] S. Pellegrinelli, A. Orlandini, N. Pedrocchi, A. Umbrico, and T. Tolio, "Motion planning and scheduling for human and industrial-robot collaboration," *CIRP Annals*, vol. 66, no. 1, pp. 1–4, 2017.

[3] M. F. Zaeh, M. Beetz, K. Shea, G. Reinhart, K. Bender, C. Lau, M. Ostgathe, W. Vogl, M. Wiesbeck, M. Engelhard, C. Ertelt, T. Rühr, M. Friedrich, and S. Herle, *The Cognitive Factory*, pp. 355–371. Springer London, 2009.

[4] "Savioke Relay." http://www.savioke.com/. Accessed: 2019-03-13.

[5] M. J.-Y. Chung, J. Huang, L. Takayama, T. Lau, and M. Cakmak, "Iterative Design of a System for Programming Socially Interactive Service Robots," in *International Conference on Social Robotics (ICSR)*, 2016.

[6] B. Hendriks, B. Meerbeek, S. Boess, S. Pauws, and M. Sonneveld, "Robot Vacuum Cleaner Personality and Behavior," *International Journal of Social Robotics*, vol. 3, pp. 187–195, Apr 2011.

[7] J. Fink, V. Bauwens, F. Kaplan, and P. Dillenbourg, "Living with a Vacuum Cleaning Robot," *International Journal of Social Robotics*, vol. 5, pp. 389–408, Aug 2013.

[8] S. Gibbs, "Google sibling Waymo launches fully autonomous ride-hailing service," *The Guardian*, Nov 11 2017.

[9] A. Dragan and S. Srinivasa, "Familiarization to robot motion," in *Proceedings of the Ninth International Conference on Human-Robot Interaction*, 2014.

[10] S. H. Huang, D. Held, P. Abbeel, and A. D. Dragan, "Enabling robots to communicate their objectives," in *Proceedings of the Thirteenth Annual Robotics: Science and Systems (RSS)*, 2017.

[11] S. H. Huang, D. Held, P. Abbeel, and A. D. Dragan, "Enabling robots to communicate their objectives," *Autonomous Robots (AURO)*, no. 2, pp. 309–326, 2019.

[12] A. Y. Ng and S. Russell, "Algorithms for inverse reinforcement learning," in *Proceedings of the Seventeenth International Conference on Machine Learning*, 2000.

[13] M. Kwon, S. H. Huang, and A. D. Dragan, "Expressing robot incapability," in *Proceedings of the Thirteenth Annual ACM/IEEE International Conference on Human Robot Interaction (HRI)*, 2018.

[14] M. Herman, T. Gindele, J. Wagner, F. Schmitt, and W. Burgard, "Inverse reinforcement learning with simultaneous estimation of rewards and dynamics," in *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2016.

[15] S. H. Huang, K. Bhatia, P. Abbeel, and A. D. Dragan, "Establishing appropriate trust via critical states," in *Proceedings of the Thirty-First Annual IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018.

[16] C. L. Baker, R. Saxe, and J. B. Tenenbaum, "Action understanding as inverse planning," *Cognition*, vol. 113, no. 3, p. 329–349, 2009.

[17] T. Ullman, C. Baker, O. Macindoe, O. Evans, N. Goodman, and J. B. Tenenbaum, "Help or hinder: Bayesian models of social goal inference," in *Proceedings of the Twenty-Second Annual Conference on Neural Information Processing Systems*, 2009.

[18] C. L. Baker, R. R. Saxe, and J. B. Tenenbaum, "Bayesian theory of mind: Modeling joint belief-desire attribution," in *In Proceedings of the Thirtieth Third Annual Conference of the Cognitive Science Society*, pp. 2469–2474, 2011.

[19] J. Jara-Ettinger, H. Gwen, L. E. Schulz, and J. B. Tenenbaum, "The naïve utility calculus: Computational principles underlying commonsense psychology," *Trends in Cognitive Sciences*, vol. 20, no. 8, p. 589–604, 2016.

[20] A. D. Dragan, K. C. T. Lee, and S. S. Srinivasa, "Legibility and predictability of robot motion," in *Proceedings of the Eighth ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pp. 301–308, ACM, 2013.

[21] J. Lasseter, "Principles of Traditional Animation Applied to 3D Computer Animation," in *Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH)*, 1987.

[22] L. Takayama, D. Dooley, and W. Ju, "Expressing thought: Improving robot readability with animation principles," in *Proceedings of the Sixth ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pp. 69–76, ACM, 2011.

[23] A. Zhou, D. Hadfield-Menell, A. Nagabandi, and A. D. Dragan, "Expressive robot motion timing," in *Proceedings of the Twelfth ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pp. 22–31, ACM, 2017.

[24] N. Sebanz, H. Bekkering, and G. Knoblich, "Joint action: Bodies and minds moving together.," *Trends in Cognitive Sciences*, vol. 10, p. 70–76, 2006.

[25] A. Dragan, S. Bauman, J. Forlizzi, and S. Srinivasa, "Effects of robot motion on human-robot collaboration," in *Proceedings of the Tenth International Conference on Human-Robot Interaction*, 2015.

[26] S. J. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*. Prentice Hall Press, 3rd ed., 2009.

[27] S. Levine and V. Koltun, "Continuous inverse optimal control with locally optimal examples," in *Proceedings of the Twenty-Ninth International Conference on Machine Learning*, 2012.

[28] D. Sadigh, S. S. Sastry, S. A. Seshia, and A. Dragan, "Information gathering actions over human internal state," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2016.

[29] S. A. Goldman and M. J. Kearns, "On the complexity of teaching," *Journal of Computer and System Sciences*, vol. 50, no. 1, p. 20–31, 1995.

[30] F. J. Balbach and T. Zeugmann, "Recent developments in algorithmic teaching," in *Proceedings of the Third International Conference on Language and Automata Theory and Applications*, 2009.

[31] A. N. Rafferty, E. Brunskill, T. L. Griffiths, and P. Shafto, "Faster teaching by POMDP planning," in *Proceedings of the Fifteenth International Conference on Artificial Intelligence in Education*, p. 280–287, 2011.

[32] S. Zilles, S. Lange, R. Holte, and M. Zinkevich, "Models of cooperative teaching and learning," *Journal of Machine Learning Research*, 2011.

[33] X. Zhu, "Machine teaching: An inverse problem to machine learning and an approach toward optimal education," in *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

[34] K. R. Koedinger, E. Brunskill, R. S. J. de Baker, E. A. McLaughlin, and J. C. Stamper, "New potentials for data-driven intelligent tutoring system development and optimization.," *AI Magazine*, vol. 34, no. 3, p. 27–41, 2013.

[35] M. Cakmak and M. Lopes, "Algorithmic and human teaching of sequential decision tasks," in *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence*, 2012.

[36] E. Vul, N. D. Goodman, T. L. Griffiths, and J. B. Tenenbaum, "One and done? Optimal decisions from very few samples," *Cognitive Science*, vol. 38, no. 4, p. 599–637, 2014.

[37] D. Hadfield-Menell, A. Dragan, P. Abbeel, and S. Russell, "Cooperative inverse reinforcement learning," in *Proceedings of the 30th International Conference on Neural Information Processing Systems (NIPS)*, 2016.

[38] D. Hadfield-Menell, S. Milli, P. Abbeel, S. Russell, and A. D. Dragan, "Inverse reward design," in *Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS)*, 2017.

[39] C. Schmidt, N. Sridharan, and J. Goodson, "The plan recognition problem: An intersection of psychology and artificial intelligence," *Artificial Intelligence*, vol. 11, no. 1, pp. 45–83, 1978.

[40] P. Cohen, C. Perrault, and J. Allen, "Beyond question answering," in *Strategies for Natural Language Processing*, pp. 245–274, 1981.

[41] D. Matsui, T. Minato, K. F. MacDorman, and H. Ishiguro, "Generating natural motion in an android by mapping human motion," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2005.

[42] M. J. Gielniak, C. K. Liu, and A. L. Thomaz, "Generating human-like motion for robots," *International Journal of Robotics Research*, vol. 32, no. 11, p. 1275–1301, 2013.

[43] M. J. Gielniak and A. L. Thomaz, "Generating anticipation in robot motion," in *20th IEEE International Symposium on Robot and Human Interactive Communication*, 2011.

[44] D. Szafir, B. Mutlu, and T. Fong, "Communication of intent in assistive free flyers," in *Proceedings of the 2014 ACM/IEEE International Conference on Human-Robot Interaction*, 2014.

[45] S. Nikolaidis, S. Nath, A. Procaccia, and S. Srinivasa, "Game-theoretic modeling of human adaptation in human-robot collaboration," in *Proceedings of the 2017 ACM/IEEE International Conference on Human-Robot Interaction*, 2017.

[46] V. Raman and H. Kress-Gazit, "Explaining impossible high-level robot behaviors," *IEEE Transactions on Robotics*, vol. 29, no. 1, p. 94–1104, 2013.

[47] S. Tellex, R. Knepper, A. Li, D. Rus, and N. Roy, "Asking for help using inverse semantics," in *Proceedings of Robotics: Science and Systems*, 2014.

[48] V. Perera, S. P. Selvaraj, S. Rosenthal, and M. M. Veloso, "Dynamic generation and refinement of robot verbalization," in *25th IEEE International Symposium on Robot and Human Interactive Communication*, 2016.

[49] S. Rosenthal, S. P. Selvaraj, and M. M. Veloso, "Verbalization: Narration of autonomous robot experience," in *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, 2016.

[50] B. Hayes and J. A. Shah, "Improving robot controller transparency through autonomous policy explanation," in *Proceedings of the 2017 ACM/IEEE International Conference on Human-Robot Interaction*, 2017.

[51] F. Khan, B. Mutlu, and X. Zhu, "How do humans teach: On curriculum learning and teaching dimension," in *Proceedings of the Twenty-Fourth Annual Conference on Neural Information Processing Systems*, 2011.

[52] S. Basu and J. Christensen, "Teaching classification boundaries to humans," in *Proceedings of the Twenty-Seventh AAAI Conference on Artificial Intelligence*, 2013.

[53] A. Singla, I. Bogunovic, G. Bartok, A. Karbasi, and A. Krause, "Near-optimally teaching the crowd to classify," in *Proceedings of the Thirty-First International Conference on Machine Learning*, 2014.

[54] K. R. Patil, X. Zhu, Ł. Kopeć, and B. C. Love, "Optimal teaching for limited-capacity human learners," in *Proceedings of the Twenty-Seventh Annual Conference on Neural Information Processing Systems*, 2014.

[55] U. Chajewska, D. Koller, and D. Ormoneit, "Learning an agent's utility function by observing behavior," in *Proceedings of the Eighteenth International Conference on Machine Learning*, pp. 35–42, 2001.

[56] D. Ramachandran and E. Amir, "Bayesian inverse reinforcement learning," in *Proceedings of the Twentieth International Joint Conference on Artifical Intelligence*, pp. 2586–2591, 2007.

[57] B. D. Ziebart, A. Maas, J. A. Bagnell, and A. Dey, "Maximum entropy inverse reinforcement learning," in *Proceedings of the Twenty-Second AAAI Conference on Artificial Intelligence*, 2008.

[58] J. Choi and K. Kim, "MAP Inference for Bayesian Inverse Reinforcement Learning," in *Proceedings of the Twenty-Fourth Annual Conference on Neural Information Processing Systems*, pp. 1989–1997, 2011.

[59] P. Abbeel and A. Y. Ng, "Apprenticeship learning via inverse reinforcement learning," in *Proceedings of the Twenty-First International Conference on Machine Learning*, 2004.

[60] R. N. Shepard, "Attention and the metric structure of the stimulus space," *Journal of Mathematical Psychology*, vol. 1, no. 1, p. 54–87, 1964.

[61] F. G. Ashby, *Multidimensional models of perception and cognition*. Psychology Press, 1st ed., 1992.

[62] H. Sakoe and S. Chiba, "Dynamic programming algorithm optimization for spoken word recognition," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 26, no. 1, pp. 43–49, 1978.

[63] G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher, "An analysis of approximations for maximizing submodular set functions," *Mathematical Programming*, vol. 14, no. 1, p. 265–294, 1978.

[64] S. Taheri and E. H. Law, "Investigation of a combined slip control braking and closed loop four wheel steering system for an automobile during combined hard braking and severe steering," in *Proceedings of the American Control Conference*, 1990.

[65] D. L. Medin and M. M. Schaffer, "Context theory of classification learning," *Psychological Review*, vol. 85, no. 3, p. 207–238, 1978.

[66] R. M. Nosofsky, "Attention, similarity, and the identification-categorization relationship," *Journal of Experimental Psychology: General*, vol. 115, no. 1, p. 39–57, 1986.

[67] D. Sadigh, S. S. Sastry, S. A. Seshia, and A. D. Dragan, "Planning for autonomous cars that leverage effects on human actions," in *Proceedings of Robotics: Science and Systems*, 2016.

[68] E. Cha, A. D. Dragan, and S. S. Srinivasa, "Perceived robot capability," in *Proceedings of the Twenty-Fourth IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*, pp. 541–548, 2015.

[69] T. Kim and P. Hinds, "Who should i blame? effects of autonomy and transparency on attributions in human-robot interaction," in *Proceedings of the Fifteenth IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*, pp. 80–85, 2006.

[70] M. N. Nicolescu and M. J. Mataric, "Learning and interacting in human-robot domains," *IEEE Trans. Systems, Man, and Cybernetics, Part A*, vol. 31, no. 5, pp. 419–430, 2001.

[71] C. J. Hayes, M. Moosaei, and L. D. Riek, "Exploring implicit human responses to robot mistakes in a learning from demonstration task," in *Proceedings of the Twenty-Fifth IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*, pp. 246–252, 2016.

[72] M. K. Lee, S. Kielser, J. Forlizzi, S. Srinivasa, and P. Rybski, "Gracefully mitigating breakdowns in robotic services," in *Proceedings of the Fifth ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pp. 203–210, ACM, 2010.

[73] D. J. Brooks, M. Begum, and H. A. Yanco, "Analysis of reactions towards failures and recovery strategies for autonomous robots," in *Proceedings of the Twenty-Fifth IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*, pp. 487–492, 2016.

[74] N. Mirnig, G. Stollnberger, M. Miksch, S. Stadler, M. Giuliani, and M. Tscheligi, "To err is robot: How humans assess and act toward an erroneous social robot," *Frontiers in Robotics and AI*, vol. 4, p. 21, 2017.

[75] M. Ragni, A. Rudenko, B. Kuhnert, and K. O. Arras, "Errare humanum est: Erroneous robots in human-robot interaction," in *Proceedings of the Twenty-Fifth IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*, pp. 501–506, 2016.

[76] E. Aronson, B. Willerman, and J. Floyd, "The effect of a pratfall on increasing interpersonal attractiveness," *Psychonomic Science*, vol. 4, no. 6, pp. 227–228, 1966.

[77] K. Kobayashi and S. Yamada, "Informing a user of robot's mind," in *Proceedings of the Third International Conference on Computational Intelligence, Robotics and Autonomous Systems (CIRAS)*, 2005.

[78] F. Stulp, J. Grizou, B. Busch, and M. Lopes, "Facilitating intention prediction for humans by optimizing robot motions," in *Proceedings of the Twenty-Eighth IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1249–1255, 2015.

[79] S. Augustsson, J. Olsson, L. G. Christiernin, and G. Bolmsjö, "How to transfer information between collaborating human operators and industrial robots in an assembly," in *Proceedings of the Eighth Nordic Conference on Human-Computer Interaction (NordiCHI): Fun, Fast, Foundational*, pp. 286–294, ACM, 2014.

[80] J. Carff, M. Johnson, E. M. El-Sheikh, and J. E. Pratt, "Human-robot team navigation in visually complex environments," in *Proceedings of the Twenty-Second IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 3043–3050, 2009.

[81] E. Rosen, D. Whitney, E. Phillips, G. Chen, J. Tompkin, G. Konidaris, and S. Tellex, "Communicating robot arm motion intent through mixed reality head-mounted displays," in *Proceedings of the International Symposium on Robotics Research (ISRR)*, 2017.

[82] B. Mutlu, F. Yamaoka, T. Kanda, H. Ishiguro, and N. Hagita, "Nonverbal leakage in robots: Communication of intentions through seemingly unintentional behavior," in *Proceedings of the Fourth ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pp. 69–76, ACM, 2009.

[83] C. Breazeal and P. Fitzpatrick, "That certain look: Social amplification of animate vision," in *AAAI Fall Symposium*, pp. 18–22, 2000.

[84] A. Haddadi, E. A. Croft, B. T. Gleeson, K. MacLean, and J. Alcazar, "Analysis of task-based gestures in human-robot interaction," in *Proceedings of the 2013 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2146–2152, 2013.

[85] B. Gleeson, K. MacLean, A. Haddadi, E. Croft, and J. Alcazar, "Gestures for industry: Intuitive human-robot communication from human observation," in *Proceedings of the Eighth ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pp. 349–356, ACM, 2013.

[86] V. Raman, C. Lignos, C. Finucane, K. C. Lee, M. Marcus, and H. Kress-Gazit, "Sorry dave, i'm afraid i can't do that: Explaining unachievable robot tasks using natural language," in *Robotics: Science and Systems*, 2013.

[87] M. Matthews, G. Chowdhary, and E. Kieson, "Intent communication between autonomous vehicles and pedestrians," *arXiv preprint arXiv:1708.07123*, 2017.

[88] M. Desai, P. Kaniarasu, M. Medvedev, A. Steinfeld, and H. Yanco, "Impact of robot failures and feedback on real-time trust," in *Proceedings of the Eighth ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pp. 251–258, IEEE Press, 2013.

[89] P. Kaniarasu, A. Steinfeld, M. Desai, and H. Yanco, "Robot confidence and trust alignment," in *Proceedings of the Eighth ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pp. 155–156, ACM, 2013.

[90] M. Zucker, N. Ratliff, A. D. Dragan, M. Pivtoraiko, M. Klingensmith, C. M. Dellin, J. A. Bagnell, and S. S. Srinivasa, "Chomp: Covariant hamiltonian optimization for motion planning," *International Journal of Robotics Research (IJRR)*, vol. 32, no. 9-10, pp. 1164–1193, 2013.

[91] J. Schulman, Y. Duan, J. Ho, A. Lee, I. Awwal, H. Bradlow, J. Pan, S. Patil, K. Goldberg, and P. Abbeel, "Motion planning with sequential convex optimization and convex collision checking," *International Journal of Robotics Research (IJRR)*, vol. 33, no. 9, pp. 1251–1270, 2014.

[92] R. Diankov, *Automated construction of robotic manipulation programs*. PhD thesis, Carnegie Mellon University, Robotics Institute, August 2010.

[93] N. Wang, D. V. Pynadath, and S. G. Hill, "Trust calibration within a human-robot team: Comparing automatically generated explanations," in *Proceedings of the Eleventh ACM/IEEE International Conference on Human Robot Interaction (HRI)*, pp. 109–116, ACM, 2016.

[94] M. T. Dzindolet, S. A. Peterson, R. A. Pomranky, L. G. Pierce, and H. P. Beck, "The role of trust in automation reliance," *International Journal of Human-Computer Studies*, vol. 58, pp. 697–718, June 2003.

[95] J. D. Lee and K. A. See, "Trust in automation: Designing for appropriate reliance," *Human Factors*, vol. 46, no. 1, pp. 50–80, 2004.

[96] S. Ososky, D. Schuster, E. Phillips, and F. Jentsch, "Building appropriate trust in human-robot teams," in *Proceedings of the Twenty-Seventh AAAI Conference on Artificial Intelligence*, 2013.

[97] A. Freedy, E. DeVisser, G. Weltman, and N. Coeyman, "Measurement of trust in human-robot collaboration," in *2007 International Symposium on Collaborative Technologies and Systems*, pp. 106–114, May 2007.

[98] B. M. Muir, "Trust between humans and machines, and the design of decision aids," *International Journal of Man-Machine Studies*, vol. 27, no. 5, pp. 527–539, 1987.

[99] D. H. McKnight and N. L. Chervany, "What trust means in e-commerce customer relationships: An interdisciplinary conceptual typology," *International Journal of Electronic Commerce*, vol. 6, no. 2, pp. 35–59, 2001.

[100] M. Lewis, K. Sycara, and P. Walker, "The role of trust in human-robot interaction," in *Foundations of Trusted Autonomy* (H. A. Abbass, J. Scholz, and D. J. Reid, eds.), pp. 135–159, Springer International Publishing, 2018.

[101] S. Levine, C. Finn, T. Darrell, and P. Abbeel, "End-to-end training of deep visuomotor policies," *Journal of Machine Learning Research*, vol. 17, no. 39, pp. 1–40, 2016.

[102] M. Bojarski, D. D. Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, X. Zhang, J. Zhao, and K. Zieba, "End to end learning for self-driving cars," *arXiv preprint arXiv:1604.07316*, 2016.

[103] B. W. Israelsen and N. R. Ahmed, "'Dave...I can assure you...that it's going to be all right..' A definition, case for, and survey of algorithmic assurances in human-autonomy trust relationships," *arXiv preprint arXiv:1711.03846*, 2017.

[104] T. Haarnoja, H. Tang, P. Abbeel, and S. Levine, "Reinforcement learning with deep energy-based policies," in *International Conference on Machine Learning*, 2017.

[105] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor.," in *Neural Information Processing Systems (NIPS) Deep Reinforcement Learning Symposium*, 2017.

[106] R. S. Sutton and A. G. Barto, *Introduction to Reinforcement Learning*. Cambridge, MA, USA: MIT Press, 1st ed., 1998.

[107] T. A. Angelo and K. P. Cross, *Classroom assessment techniques: A handbook for college teachers*. Jossey-Bass Publishers, 2nd ed., 1993.

[108] J. M. Webb, E. M. Diana, P. Luft, E. W. Brooks, and E. L. Brennan, "Influence of Pedagogical Expertise and Feedback on Assessing Student Comprehension from Nonverbal Behavior," *The Journal of Educational Research*, vol. 91, no. 2, pp. 89–97, 1997.

[109] B. D. Argall, S. Chernova, M. Veloso, and B. Browning, "A survey of robot learning from demonstration," *Robotics and Autonomous Systems*, vol. 57, no. 5, pp. 469–483, 2009.

[110] M. Cakmak and A. L. Thomaz, "Optimality of human teachers for robot learners," in *Proceedings of the IEEE Ninth International Conference on Development and Learning (ICDL)*, 2010.

[111] A. Sena, Y. Zhao, and M. J. Howard, "Teaching human teachers to teach robot learners," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2018.

[112] T. Inamura, M. Inaba, and H. Inoue, "Acquisition of probabilistic behavior decision model based on the interactive teaching method," in *Proceedings of the Ninth International Conference on Advanced Robotics (ICAR)*, 1999.

[113] A. P. Shon, D. Verma, and R. P. N. Rao, "Active imitation learning," in *Proceedings of the 22nd National Conference on Artificial Intelligence (AAAI)*, 2007.

[114] S. Chernova and M. Veloso, "Interactive policy learning through confidence-based autonomy," *Journal of Artificial Intelligence Research*, vol. 34, no. 1, pp. 1–25, 2009.

[115] D. Silver, J. A. Bagnell, and A. Stentz, "Active learning from demonstration for robust autonomous navigation," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2012.

[116] J. Miura, K. Iwase, and Y. Shirai, "Interactive teaching of a mobile robot," in *Proceedings of the 2005 IEEE International Conference on Robotics and Automation (ICRA)*, 2005.

[117] D. Whitney, E. Rosen, J. MacGlashan, L. L. S. Wong, and S. Tellex, "Reducing errors in object-fetching interactions through social feedback," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, 2017.

[118] C. Chao, M. Cakmak, and A. L. Thomaz, "Transparent active learning for robots," in *Proceedings of the Fifth ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, 2010.

[119] M. Cakmak, C. Chao, and A. L. Thomaz, "Designing interactions for robot active learners," *IEEE Transactions on Autonomous Mental Development*, vol. 2, pp. 108–118, June 2010.

[120] S. Calinon and A. Billard, "Active teaching in robot programming by demonstration," in *Proceedings of the IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*, 2007.

[121] L. Steels and F. Kaplan, "AIBO's first words: The social learning of language and meaning," *Evolution of Communication*, vol. 4, no. 1, pp. 3–32, 2001.

[122] G. H. Lim, M. Oliveira, V. Mokhtari, S. H. Kasaei, A. Chauhan, L. S. Lopes, and A. M. Tomé, "Interactive teaching and experience extraction for learning about objects and robot activities," in *The 23rd IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*, 2014.

[123] H. Admoni and B. Scassellati, "Social eye gaze in human-robot interaction: A review," *Journal of Human-Robot Interaction*, vol. 6, no. 1, pp. 25–63, 2017.

[124] H. Admoni, T. Weng, and B. Scassellati, "Modeling communicative behaviors for object references in human-robot interaction," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2016.

[125] J.-D. Boucher, U. Pattacini, A. Lelong, G. Bailly, F. Elisei, S. Fagel, P. F. Dominey, and J. Ventre-Dominey, "I reach faster when I see you look: Gaze effects in human–human and human–robot face-to-face cooperation," *Frontiers in Neurorobotics*, vol. 6, no. 3, 2012.

[126] A. Moon, D. M. Troniak, B. Gleeson, M. K. X. J. Pan, M. Zheng, B. A. Blumer, K. MacLean, and E. A. Croft, "Meet me where I'm gazing: How shared attention gaze affects human-robot handover timing," in *Proceedings of the 2014 ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, 2014.

[127] H. Admoni, A. Dragan, S. Srinivasa, and B. Scassellati, "Deliberate Delays During Robot-to-Human Handovers Improve Compliance With Gaze Communication," in *International Conference on Human-Robot Interaction (HRI)*, 2014.

[128] A. L. Thomaz and C. Breazeal, "Teachable robots: Understanding human teaching behavior to build more effective robot learners," *Artificial Intelligence*, vol. 172, no. 6-7, pp. 716–737, 2008.

[129] D. A. Pomerleau, "Efficient training of artificial neural networks for autonomous navigation," *Neural Computation*, vol. 3, pp. 88–97, March 1991.

[130] S. Ross, G. J. Gordon, and J. A. Bagnell, "A reduction of imitation learning and structured prediction to no-regret online learning," in *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2011.

[131] P. A. Ortega and A. A. Stocker, "Human decision-making under limited time," in *Advances in Neural Information Processing Systems (NeurIPS) 29*, 2016.

[132] N. J. Gordon, D. J. Salmond, and A. F. M. Smith, "Novel approach to nonlinear/non-Gaussian Bayesian state estimation," *IEE Proceedings F - Radar and Signal Processing*, vol. 140, pp. 107–113, April 1993.

[133] M. Nowak and K. Sigmund, "A strategy of win-stay, lose-shift that outperforms tit-for-tat in the Prisoner's Dilemma game," *Nature*, vol. 364, pp. 56–58, 1993.