Robot See, Robot Do: On the Development of Robust and Adaptive Imitation Learning for Robots

Shaunak A. Mehta

Dissertation submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy
in
Mechanical Engineering

Dylan P. Losey, Chair
Michael D. Bartlett
Kaveh Akbari Hamed
Andrea Bajcsy

September 3, 2025
Blacksburg, Virginia

Keywords: Imitation Learning, Human-Robot Interaction, Learning Dynamics

Robot See, Robot Do: On the Development of Robust and Adaptive Imitation Learning for Robots

Shaunak A. Mehta

(ABSTRACT)

As robots transition from isolated industrial settings to working in close proximity to humans in dynamic environments, their ability to learn and adapt to human feedback and unseen circumstances becomes crucial. Imitation learning offers a promising paradigm for robots to learn complex tasks by mimicking human behavior. However, traditional imitation learning approaches face key challenges in integrating diverse feedback types, managing noisy and inconsistent inputs, and maintaining stability in learning. In this thesis, we develop imitation learning approaches that advance the capabilities of robots and enable them to efficiently learn from humans and adapt to unseen data in diverse environments.

This research is structured around four key contributions. First, we consider the scenario where a human is readily available to provide high-quality feedback to the robot. We develop a learning algorithm to enable robots to learn from diverse sources of optimal human feedback: demonstrations, corrections, and preferences. Demonstrations provide high-level task overviews, corrections fine-tune specific motions, and preferences rank robot behaviors for task improvement. By incorporating these active and passive feedback sources under a unified reward learning framework we enable robots to infer task objectives more effectively and optimize their trajectories using constrained optimization techniques.

Second, we explore scenarios where the human feedback is noisy or biased due to task complexity or physical constraints. We model the robot's learning rule as a dynamical system and apply Lyapunov stability analysis to derive conditions of convergence. Leveraging these

conditions, we modify the robot's learning rule to expand the basins of attractions around the possible tasks (equilibrium points) in the environment. This approach enables the robot to infer the correct task representations from a wider range of human inputs making the learning robust to suboptimal feedback without destabilizing the robot behavior.

Next, we consider imitation learning settings where a human is not available to provide additional feedback. In such scenarios, imitation learning algorithms are often prone to covariate shift when they encounter data not seen during training. To tackle this challenge, we develop Stable Behavior Cloning (Stable-BC), a stability-driven imitation learning algorithm. This algorithm ensures that robots maintain reliable performance by encouraging policy stability around demonstrated behaviors without the need for additional training data or complex reinforcement learning methods.

Finally, we look at the problem of imitation learning from the users' perspective and aim to reduce the time and effort required to teach the robot. We propose L2D2, a sketching interface and imitation learning algorithm where humans can provide demonstrations by drawing the task. L2D2 leverages vision-language segmentation to autonomously vary object locations and generates synthetic images of the environment for the human to draw upon. By collecting a few physical demonstrations from the users, L2D2 then grounds these diverse 2D drawings in the real world. This approach reduces the time and effort required to teach the robots by enabling the users to rapidly provide a large set of diverse demonstrations.

The findings from this research highlight the importance of adaptability and stability when robots and autonomous agents work around and interact with humans in diverse environments. This research contributes to the broader field of robot learning by offering scalable, adaptable, and user-friendly solutions for imitation learning and human-robot interaction, paving the way for more intuitive and robust robotic systems in human environments.

Robot See, Robot Do: On the Development of Robust and Adaptive Imitation Learning for Robots

Shaunak A. Mehta

(GENERAL AUDIENCE ABSTRACT)

As robots move beyond factory floors and start working alongside people in homes and workplaces, it becomes important for them to learn from human instructions and handle unexpected situations. One way robots can learn is by observing and copying human actions, similar to how children learn by watching adults. This approach is called imitation learning. While promising, it has challenges: robots need to handle different types of human instructions, deal with unclear or messy guidance, and keep their learning stable over time without making errors. This research focuses on creating smarter and more adaptable learning methods for robots to address these challenges.

This research is organized around four main contributions. The first part of the research looks at situations where humans are available to give helpful guidance to robots. We develop a method that allows robots to learn from three types of human input: demonstrations (showing how to do a task), corrections (fixing mistakes), and preferences (choosing which robot action is better). By combining these types of input, robots can better understand what humans want and adjust their actions to complete tasks successfully.

In the second part, we explore situations where human instructions are not perfect, perhaps due to the complexity of the task or physical limitations that affect their inputs. Sometimes, humans make mistakes or provide unclear feedback. To address this, we treat the robot's learning process as a system that needs to stay balanced and stable, even when the guidance is noisy. We develop a method that helps the robot understand the correct task from a wider

range of human inputs without misinterpreting the inputs or becoming unstable.

Next, we consider cases where a human isn't available to provide additional guidance during the robot's learning process. In such cases, robots often struggle when they face situations they haven't seen before. This can lead to the robot taking incorrect actions resulting in errors that get worse over time. To solve this problem, we create an approach called Stable Behavior Cloning, which helps robots stay consistent and perform reliably without needing additional training or complex trial-and-error learning.

Finally, we think about the problem from the human user's point of view and try to make it easier and faster to teach robots. We introduce L2D2, a tool where people can teach robots by simply drawing what they want the robot to do. L2D2 uses advanced computer vision and language techniques to create different versions of the task environment automatically, so people can quickly provide many different examples. This makes teaching robots faster and less work for the user.

This research shows that adaptability and stability are essential for robots to work safely and effectively alongside humans in diverse, real-world environments. The findings of this research help us move closer to building robots that are easier to use, safer, and more helpful in everyday life.

Acknowledgments

I am incredibly fortunate to have received the support of so many people on this journey, and I am deeply grateful for their contributions, which have made reaching this stage possible. First and foremost, I would like to express my sincere gratitude to my advisor, Dr. Dylan P. Losey, for his unwavering guidance and invaluable research insights, without which this thesis would not have been possible. I am also deeply thankful to my committee members, Dr. Michael D. Bartlett, Dr. Kaveh Akbari Hamed and Dr. Andrea Bajcsy, for their time, effort, and insightful feedback that significantly enriched this work. A heartfelt thank you to my labmates — Ananth, Soheil, Sagar, Shahab, Heramb, and Ben — for the countless brainstorming sessions, support, and camaraderie during social events that helped me navigate this stage of life. I owe a profound debt of gratitude to my family, especially my parents, grandmother, and the cherished memory of my late grandfather. Their unwavering love, hard work, and constant support have been the foundation upon which I stand today. Lastly, I extend my deepest thanks to my roommates and friends, whose companionship and encouragement kept me motivated to reach the end of this wonderful journey.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Traditionally, robots were designed to perform repetitive and labor-intensive tasks in controlled environments, such as manufacturing and assembly lines, where they operated in isolation from humans. Over time, these robots and autonomous agents have evolved, transitioning from controlled environments to human-centric settings where they work in close proximity to people. Examples of such environments include household tasks [7, 8], collaborative activities where humans and robots jointly accomplish goals [9], and driving alongside human users[10, 11]. As robots increasingly integrate into these dynamic spaces, their ability to learn and adapt to new tasks becomes essential for successful deployment. This raises a fundamental question: How can robots efficiently learn to perform such diverse tasks in dynamic, human-centered environments?

To explore this question, we can draw inspiration from how humans learn. Imagine a child learning to pick up a cup. While there are many ways to grasp the cup, some methods—such as holding it upside down—may cause the contents to spill. Typically, an adult demonstrates the correct method, and the child attempts to imitate these actions to perform the task successfully [12, 13]. If the child makes a mistake when performing the task, the adult can intervene and correct them. This illustrates that one effective way to learn diverse tasks is through imitation of an expert by incorporating their feedback. Similarly, robots can learn from human experts by mimicking their actions and behaviors [14].

Imitation learning has emerged as a foundational paradigm in robot learning, enabling robots

to acquire complex skills by imitating human-provided demonstrations. This approach is particularly appealing due to its intuitive nature: humans demonstrate the desired task, and robots learn by replicating the observed behavior. For instance, a robot or an autonomous agent can be taught to perform tasks such as driving, assembling parts, playing interactive games, or navigating dynamic environments through demonstrations. However, despite its promise, imitation learning faces significant challenges in effectively learning from diverse human inputs in real world. Overcoming these challenges requires a shift in focus from traditional approaches to develop more robust, adaptive, and versatile frameworks to accommodate diverse feedback modalities, imperfect human inputs, and unseen task scenarios.

One of the major challenges in imitation learning is real-time learning from diverse forms of human feedback. As robots interact with humans, their behavior is often guided or corrected on the fly through physical interventions [15, 16], gestures [17], or verbal feedback [18, 19]. While imitation learning traditionally relies on a single form of feedback, real-world human-robot interaction (HRI) often demands that robots adapt their behavior in real time from diverse forms human inputs. For example, a robot delivering objects in a cluttered environment may receive corrections to avoid specific obstacles or feedback about user preferences for delivery paths. Such inputs require the robot to incorporate feedback quickly and generalize from it to improve future performance.

Another challenge faced by imitation learning is learning from diverse qualities of the human provided inputs. Since humans are not perfect, their inputs itself are often noisy, inconsistent, or suboptimal [1]. Humans may over-correct a robot's motion, provide conflicting instructions, or unintentionally introduce errors in their guidance. Traditional imitation learning methods lack mechanisms to distinguish and stabilize learning under such conditions, leading to unpredictable behavior or failures to converge to the desired outcome. The inability to robustly integrate imperfect feedback into the learning process limits the

effectiveness of robots in collaborative or dynamic settings.

When a human is not available to provide corrections or additional feedback to the robot, imitation learning is constrained by its reliance on limited and often incomplete training datasets. Robots trained through demonstrations are constrained by the scope and variability of the examples that are provided. When encountering states that fall outside this dataset during execution, their learned policies often fail, leading to errors that compound over time. Known as covariate shift, this phenomenon is particularly problematic in dynamic environments where unexpected situations are inevitable [20, 21]. Efforts to mitigate this issue by expanding datasets [22, 23] or involving humans in real-time corrections [16] are resource-intensive and do not guarantee comprehensive coverage of possible states.

While these approaches aim to learn robust policies from diverse human inputs, they often require repeated physical interactions of the user with the robot or its environment while providing demonstrations. This makes the process of teaching robots expensive in terms of human time and effort. To tackle this, recent approaches facilitate information-rich demonstrations without direct physical contact, such as through remote teleoperation leveraging physical twins [24, 25] and virtual reality [26], or by video-based task recordings [27, 28]. However, these approaches still require full task execution and environment resets, constraining data collection efficiency. In contrast to these solutions, an emerging alternative involves providing demonstrations via drawings on images of the environment [29, 30], thereby significantly reducing human effort. However, this approach inherits challenges common to traditional imitation learning, including ambiguity in interpreting sketches and difficulty in ensuring coverage of relevant state variations.

In this thesis, we tackle these fundamental challenges in imitation learning to equip robots with the tools needed to operate reliably in dynamic, human-centered environments. Our primary focus is enabling robots to learn from diverse forms of human feedback in real

time, improving stability when learning from suboptimal, noisy, and incomplete datasets, and reducing the physical burden on users during teaching. Through this work, we strive to advance the capabilities of robots, fostering their successful integration into complex, real-world applications. Overall, this thesis makes the following contributions:

Unified Learning from Demonstrations, Corrections, and Preferences during Physical Human-Robot Interaction. We present a learning approach to physical interaction that unifies demonstrations, corrections, and preferences. Our approach is based on the insight that the human's inputs are better than the alternatives: e.g., the human's demonstrated trajectory should receive higher reward than noisy perturbations of that same trajectory. Our learning approach automatically generates trajectory deformations of the human's physical interactions, and then learns to assign higher rewards to the human's actual behavior. We incorporate learning from both active and passive sources of feedback into a flexible reward learning framework. In a setting where the human may use multiple forms of feedback to teach the robot, we also enable the robot to actively prompt the human by eliciting their preferences. We identify a method for generating preference trajectories that — when ranked by the human — minimizes the robot's uncertainty over its learned reward. This approach results in an end-to-end model of the human's reward function. We then harness off-the-shelf optimization techniques and the robot's underlying kinematics to convert this learned reward function into a desired robot trajectory. We compare our approach to state-of-the-art baselines across experiments with real robot arms and simulated and real human users. First, we consider approaches for physical human-robot interaction that learn from either demonstrations and corrections or demonstrations and preferences, and show that our method more accurately learns the human's task (particularly when this task is new or unexpected). Second, we perform a user study with imitation learning baselines that synthesize multiple forms of human feedback. Here we show that participants prefer working

with robots that learn using our approach, and that our approach learns trajectories that better align with the human's intended tasks. This paper has been published in [31]

Stabilized and Robust Online Learning from Humans. We write real-time learning from human feedback as a dynamical system where the human's true preferences are the equilibrium point. We then apply Lyapunov stability analysis to derive the conditions for converging to this equilibrium. We introduce an approach that modifies the robot's learning rule to be more robust-by-design. Given a prior over human preferences and/or a human model, the robot shapes the learning rule to increase the basins of attraction and converge under a larger set of human inputs. We refer to the resulting algorithm as **StROL**: **St**abilized and **R**obust **O**nline **L**earning. We perform simulations and a user study across scenarios with robot arms and autonomous driving. We demonstrate that the learning rules produced by StROL are more robust to noisy and suboptimal humans than state-of-the-art alternatives. This work has been published in [32]

Controlling Covariate Shift with Stable Behavior Cloning. We write covariate shift as a linearized dynamical system. We then use these error dynamics to derive model-based and model-free stability conditions. These conditions bound the covariate shift in the robot state as a function of the covariate shift in the environment state. We leverage our analysis to develop Stable-BC, a behavior cloning algorithm that encourages policy stability around the dataset. Two key advantages of our approach are (a) it is easy to implement and (b) it does not alter the training data. As such, Stable-BC can be applied independently or alongside existing on-policy or off-policy methods. We conduct simulated experiments in interactive, nonlinear, and visual environments. We also perform a real-world study where human users teach a robot arm to play a simplified game of air hockey. Across all experiments, we find that Stable-BC results in more robust and effective policies than state-of-the-art baselines. This work has been published in [33]

Robot Learning from 2D Drawings. We create an interface that people can use to provide information-rich sketches to teach the robot. This interface presents a third-person image of the environment, and enables users to draw the robot's trajectory, annotate where the robot should rotate, open or close its gripper, and also highlight environment features that may vary at test time. We analyze the process of decoding task sketches into real robot trajectories using Principal Component Analysis (PCA). We model the information loss as a function of the viewpoint from which we capture the images that users draw on. Leveraging this model, we propose a strategy for optimally placing the camera to minimize information loss, and establish an upper bound on this loss when we use non-linear mappings between 3D robot trajectories and 2D sketches. Given the importance of varied examples in imitation learning and the difficulty of collecting them physically, we leverage vision-language models to synthetically augment the data collection process. Specifically, we enable users to identify relevant features (e.g., target objects) through text prompts and autonomously vary those features with each drawing, resulting in a diverse training dataset without physically manipulating the environment. Recognizing that sketches themselves are not sufficient to learn dynamic, high-dimensional tasks due to information gaps in static, low-dimensional sketches, we utilize a few physical corrections to ground the synthetic sketches in the physical environment to learn the dynamic task interactions more precisely. Our resulting approach, L2D2, integrates drawings and physical feedback in a way that amplifies their complementary strengths: using physical corrections to improve sketch accuracy, resulting in data that is both varied and reliable. We conduct real-world experiments with expert and novice users to compare our approach to state-of-the-art approaches in learning from drawings and standard baselines that learn from physical demonstrations. Our results show that the data collected by L2D2 enhances the robot's task performance compared to alternative sketching methods, while requiring significantly less time and effort as compared to traditional teleoperation approaches. This work has been accepted for publication [34].

Chapter 2

Review of Literature

In this chapter, we provide an overview of the existing research and discuss state-of-the-art approaches that tackle different challenges in the broad domain of imitation learning — learning from diverse forms of physical feedback, robust online learning from humans, stability in imitation learning, and learning from drawings. For each domain, we summarize prior works, highlight their limitations, and describe how our proposed approaches build upon or diverge from existing methods to tackle the identified challenges. First, in Section 2.1, we review recent methods for physical human–robot interaction and approaches that combine multiple sources of physical feedback. Section 2.2 focuses on real-time learning from humans, examining how to model human–robot interaction as a dynamical system to better capture human preferences. In Section 2.3, we summarize the recent methods for robust behavior cloning, as well as works that apply stability analysis in similar imitation learning settings. Finally, Section 2.4 explores different interfaces and data augmentation techniques that reduce the effort required to provide rich demonstrations, including approaches that leverage drawings as a form of feedback.

2.1 Learning from Diverse Forms of Physical Feedback

Control Responses for Physical Interaction. Although we will focus on learning, we recognize that prior research has also explored control theoretic responses to physical interac-

tion [35, 36, 37, 38, 39, 40, 41, 42]. Works on impedance control and shared control arbitrate leader-and-follower roles between the human and robot: when the human intervenes and applies large forces to the robot, the robot reduces its control feedback so that the human backdrives the arm. Intuitively, the robot can also pause, move away from the human, or follow the human's motion during physical interaction [39]. These control responses are an important step towards safety, and we will leverage shared control throughout Chapter 3 to enable close physical interaction.

Learning Rewards during Physical Interaction. Recent work has enabled robots to learn from physical human-robot interaction [43]. Here the human physically pushes, pulls, and twists the robot arm, and the robot attempts to infer why the human is applying these forces so that it can update its behavior accordingly. Some works directly map the human's forces and torques to changes in the robot's desired trajectory [44, 45, 46, 47]. But more relevant here is research that infers a reward function from physical interaction: these approaches assume that the human has in mind an objective, and the human's interactions are observations of this latent reward function [48, 49, 50]. The learned reward is then used to identify the robot's desired trajectory.

In practice, these approaches often take advantage of physical human corrections [1, 2, 3, 51, 52, 53]. Using the insight that the human's applied forces and torques are an intentional improvement — i.e., the human is going out of their way to show the robot a better way to perform the task — the robot learns to give the human's behavior higher rewards and propagate those changes the next time the robot repeats this task. In Chapter 3 we leverage a similar insight to derive our learning algorithm. However, while prior works for physical human-robot interaction focus on a single input modality (e.g., physical corrections), we will develop a framework that incorporates the different aspects of physical feedback.

Combining Multiple Feedback Modalities. Outside of physical interaction several

methods have been proposed to learn from different types of human feedback. For example, interactive imitation learning approaches can synthesize demonstrations and corrections [15, 16, 54, 55]. Consider a human watching a mobile robot: first the human might teleoperate the robot throughout several laps of the building, and then the human may jump in and correct the robot only when it makes a specific mistake. Alternatively, methods that learn from suboptimal humans often combine demonstrations and preferences [56, 57, 58, 59, 60, 61]. Imagine that you are teaching a simulated robot to play an Atari game. After you do your best to provide a demonstration — and score as high as possible — you can rank the robot's autonomous performance to indicate when it is performing well and when it is making mistakes. Most relevant to our research are [62] and [4], where the authors unite different types of human feedback to learn a single reward model.

When applying these existing approaches to physical human-robot interaction we are faced with two problems. On the one hand, methods like [4, 62] require prior knowledge about the human's reward function — as we will show in our analysis and experiments, these methods fall short when the human wants to teach the robot a new or unexpected task. On the other hand, approaches like [56, 57, 58, 59, 60, 63] use reinforcement learning to convert the reward function into robot behavior. But reinforcement learning is time consuming and requires trial and error — which may not be possible (or safe) when humans are physically interacting with robot arms. Accordingly, in Chapter 3 we introduce a formalism that unites demonstrations, corrections, and preferences without the need for pre-defined tasks or reinforcement learning.

2.2 Robust Online Learning from Humans

Continual Learning from Human Inputs. Online learning explores how robots can infer preferences from humans in real-time. Prior works have applied online learning from human

feedback to autonomous vehicles [15], assistive exoskeletons [64], and robot arms [65]. But to enable rapid adaptation, online learning often requires simplifying assumptions. Relevant works like [2, 5, 6, 31, 45, 47, 66, 67] maintain a point estimate of what the human wants, and update this estimate using gradient descent. Unfortunately, the approximations needed for online learning also make the system sensitive to suboptimal human inputs. When the user inevitably makes a mistake (and incorrectly intervenes) the robot may learn the wrong preferences [5] or misrepresent the human's true intentions [6]. Instead of thinking of this as a *learning* problem, we instead treat this as a *control* problem: how should robots modify their learning rule to ensure effective performance across suboptimal human inputs?

Learning from Humans as a Dynamical System. As a step towards fast and seamless adaptation, we will model online robot learning from humans as a *dynamical system*. Recent works have found different ways to incorporate learning mechanisms into dynamics models of human-robot interaction. This includes shared control settings where the robot adjusts its desired trajectory based on applied forces and torques [40, 46], jointly learning a model of the human policy and physical dynamics [68], modeling the human's learning process as a dynamical system [69], and dynamic movement primitives that react to human behaviors [70]. Across many of these previous works, the authors propose a learning rule, and then apply control theory to check if the resulting dynamics are stable. In Chapter 4 we take the opposite perspective. We first identify the conditions for stability, and then modify the learning rule so that it satisfies these conditions for as many human inputs as possible (i.e., we use control theory to design the learning rule).

Priors over Human Preferences. A core idea for our approach is that — even when the robot is learning in continuous spaces — there are distinct *modes* of human preferences. Think back to our motivating example: there is a continuous spectrum of distances between the robot and pitcher that the human could prefer. However, at a high level, the space

of preferences can be divided into two distinct preference modes: avoiding the pitcher or ignoring it altogether. Research in cognitive science and machine learning suggests that humans have *strong priors* over how other people with act [71, 72] and what sort of behaviors are reasonable [73, 74]. Robots can often obtain these priors from data: recent works have shown that large language models accurately predict the different actions a human might take [75]. Building on these works, we leverage intuitive, multimodal priors over the continuous preference space to shape the robot's learning rule, yielding more robust and efficient learning from human data.

2.3 Stability in Imitation Learning

Off-Policy Approaches. Within behavior cloning [76], the robot is given a dataset of expert state-action pairs and learns a policy to match this dataset. One way to enhance the robustness of the learned policy is leveraging Offline-RL [77, 78], which trains a policy to optimize a reward function in addition to matching the human demonstrations. However, a reward function in addition to the human demonstrations may not always be available. In the absence of a reward function the robustness of the learned policy can be enhanced by improving the quality of the original dataset. For example, some off-policy methods intentionally perturb the human when they are providing demonstrations, and then record how the human corrects the robot in response to those disturbances [79, 80]. Other approaches synthetically augment the dataset provided by the human to enhance the diversity and quantity of training data [22, 23, 81, 82, 83]. For example, Ke *et al.* [22] learn a dynamics model from the data, and then use that model to generate new state-action pairs that align with the expert's demonstrations.

On-Policy Approaches. The robot can also iteratively gather new data by executing its

policy in the environment, and then asking the human to provide expert guidance when the robot makes mistakes [16]. Recent works in interactive imitation learning explore when to query the human for additional data, and how to best leverage that data to fine-tune the learned policy and accelerate the robot's learning [15, 31, 54, 55, 84, 85, 86]. For example, in Menda *et al.* [84] the robot maintains an ensemble of behavior cloned models; the robot asks for human guidance when it encounters a state where the models disagree over which action to take. , while in Zhang *et al.* [85] the robot pretrains a policy with offline demonstrations followed by online fine-tuning.

Overall, both off-policy and on-policy methods focus on the data used to train the behavior cloned agent. This is orthogonal to our approach, where we will only consider the control theoretic properties of the learned policy.

Stability-Centric Approaches. Multiple related works have applied stability analysis to imitation learning. For example, safety filters [87, 88, 89] monitor the robot's real-time error (e.g., covariate shift), and revert to a given or learned backup policy when that error exceeds some threshold. Other methods structure the robot's policy like a dynamical system, and ensure that this system converges towards a goal state [32, 90, 91]. The robot can also take advantage of the stability of the human teacher: Pfrommer *et al.* [92] train the robot's behavior cloned policy to match the higher order derivatives of the human's demonstrations.

Our approach is most similar to recent research by Kang *et al.* [93]. In [93] the authors seek to prevent distribution shift in learned policies by formulating the entire training distribution as an equilibrium point. However, [93] focuses on model-based reinforcement learning — by contrast, in Chapter 5, we study local stability for behavior cloning.

2.4 Learning from Drawings

Data in Imitation Learning. A widely used imitation learning method in robotics is behavior cloning (BC) [33, 76] in which the robot is leverages a dataset of observation-action pairs to learn a policy for some task. A key factor in the effectiveness of these approaches is the quality and variety of human data [94]. For example, to learn how to pick up a cup from various locations, the robot must be shown how an expert performs the task in at least a few different positions. But obtaining this data can require significant human effort in setting up the environment and demonstrating the task [95]. Previous research has attempted to address this problem from different angles: making it easy for humans to demonstrate the task [96], synthetically augmenting the human's data [97], and developing data-efficient learning algorithms [98]. Additionally, parameterized task learning has been explored to enable the robot to generalize to different task conditions by leveraging the task-specific parameters to learn diverse robot behaviors [99, 100, 101]. In Chapter 6, we utilize existing learning rules and investigate frameworks for obtaining the required training examples with novel interfaces and data augmentation techniques.

Interfaces for Providing Demonstrations. Humans can demonstrate their desired task by guiding the robot through the task [25, 26, 31, 102, 103] or by performing the task themselves [27, 104]. Each approach has its own advantages; while performing the task directly is more natural for humans, controlling the robot results in data that is easier for the robot to learn from. Below, we discuss these approaches in more detail.

Traditional approaches for guiding the robot include kinesthetically moving the robot [43] or teleoperating the robot using a joystick [9, 102] or a space mouse [103]. While these interfaces are easily accessible, they are not always intuitive to use when controlling high-dimensional robots. To make this process more intuitive, recent works have developed in-

telligent mappings for existing interfaces [105] as well as new tools like dexterous robot arm copies [24, 25, 96]. Some works have also explored the use of virtual reality systems for teleoperating the robot from an egocentric view [26, 106, 107] and to enable the end-users to demonstrate the tasks without access to a physical robot [108, 109].

Humans do not need to consider these interfaces when performing the task themselves. Robots can learn to perform diverse tasks by observing the human's behavior from a third-person camera [27, 28, 110, 111]. However, since this data does not include observations and actions from the robot's viewpoint, it needs to be processed to extract relevant data. For this reason, other approaches have developed reacher-grabber tools equipped with onboard cameras that humans can hold to perform tasks from the robot's perspective [104, 112, 113].

A limiting factor of both these teaching modalities is that they require humans to either directly or vicariously interact with the physical environment. Further, despite the method employed to obtain demonstrations, humans still need to manually vary the environment in order to introduce diversity in the training data.

Data Augmentation. To acquire sufficient data with only a few physical interventions, robots can leverage open-source videos of humans performing the task [114] or large-scale cross-domain robot datasets [115, 116, 117], and then transfer their knowledge to the desired task [98, 118, 119, 120]. Recent advances have also utilized large language models to facilitate this transfer through contextual language prompts [121, 122, 123]. Alternatively, robots can collect a small amount of data from humans and augment it synthetically — either by adding noise to human actions and then taking corrective actions to return to the demonstrated states [79, 124], or by introducing semantic variations in the robot's observation [125, 126, 127], or both [97]. For instance, robots can employ pre-trained vision-language models to segment out regions of interest in images of the environment, and in-paint those regions with different objects and backgrounds to make the policy robust to scene changes [126].

While these approaches help the robot generalize to unseen but familiar tasks and adapt to minor changes in behaviors and visual scenes, they do not generate new expert data. For example, the robot may learn to pick the cup from the same position across visually different environments, it would not be able to pick the cup from a new location that requires a distinct motion.

Learning from Sketches. In Chapter 6, we propose leveraging sketches to quickly generate diverse data for training robots. Upcoming research in imitation learning has explored collecting demonstrations in the form of sketches of the desired behavior [29, 30, 128, 129, 130, 131]. In these approaches, users see an image of the environment and provide drawings that demonstrate the task. These drawings are then mapped to real robot trajectories [130] or used to condition the robot's behavior [30]. For instance, [29] collects 2D trajectories drawn on images taken from two viewpoints and generates the corresponding 3D trajectory using a pre-trained autoencoder framework, while [30] uses the drawing as input to the robot's policy and outputs the actions for rolling out the trajectory in the real world.

While sketches offer an intuitive way for humans to demonstrate the desired task without physically interacting with the environment, previous work has limited users to drawing on only real-world images, requiring them to reset the environment and collect new images before providing new sketches for the task. In Chapter 6, we recognize the potential of using sketches to rapidly generate large amounts of diverse training data and apply insights from data augmentation to synthetically create varied images of the task. We then obtain expert drawings on these synthetic images, capturing distinct task behaviors. Recognizing the limitations of sketches in conveying precise and dexterous motions, we ground them with a small number of physical demonstrations to produce a dataset that is both varied and high quality to enable the users to teach the robot more efficiently as compared to the state-of-the-art approaches for learning from drawings.

Chapter 3

Unifying Demonstrations, Corrections and Preferences for Robot Learning

First, we consider the case where the robots learn online from human feedback. This scenario is particularly relevant when the human and robot are working in close proximity or a human needs to teach a task to the robot where the human needs analyze the robot's behavior and provide inputs to correct or improve its behavior. In such settings, the flexibility to choose the type of feedback — demonstrations, corrections or preference queries — can help the human to effectively convey the correct task to the robot. In this chapter, we introduce an algorithm that can combine these multiple sources of human feedback under a unified framework. This formulation enables the human user to provide relevant inputs to the robot while also enabling the robot to query the human in parts of task where it is uncertain, thus leading to a successful learning of the task.

3.1 Introduction

Imagine teaching a robot arm on a factory floor (see Figure 3.1). You know what task you want the robot to perform — attaching a chair leg — but you do not know how to program the robot to perform this task. Instead, you *physically* interact with the robot. Perhaps you start by kinesthetically guiding the robot across a full demonstration of the task. Next

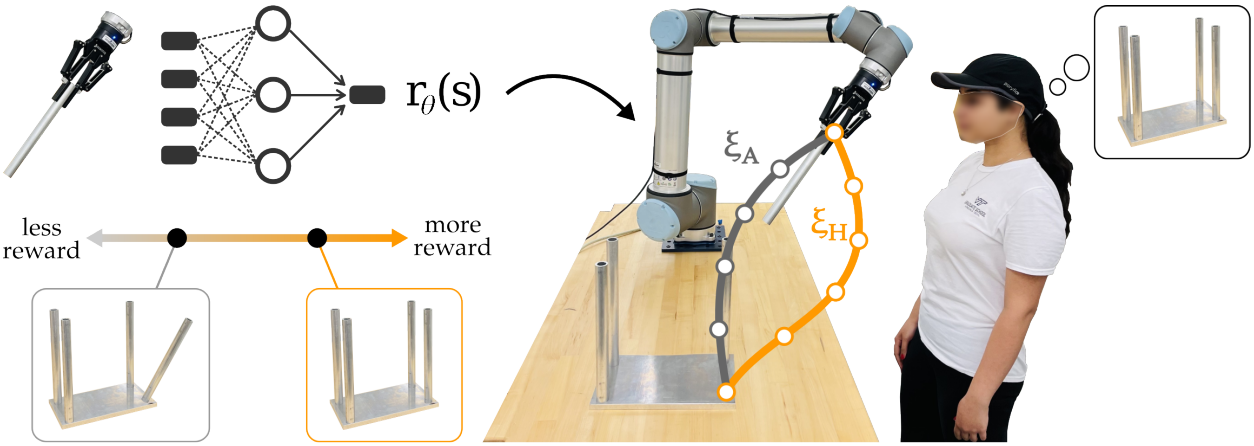


Figure 3.1: Human teaching a robot arm to assemble a chair. The robot does not have any prior information about this task, and must learn from the human's physical interactions. We recognize that these interactions can take multiple different forms, including demonstrations, corrections, and preferences. To unify each type of input under a single framework, we train a reward model to assign higher scores to the human's behavior (ξ_R) than to nearby alternatives (ξ_A). The robot then optimizes this reward model to find its desired trajectory.

you deploy the robot, and notice it is attaching the chair leg at the wrong angle: here you physically intervene and correct just that part of the arm's motion. Finally, as the robot gets closer to understanding your task, you might rank the robot's behavior, indicating times where it gets it right and times where it messes up. Throughout this process it does not make sense for you to be constrained to only a single type of interaction (e.g., only ever showing demonstrations of the task). Instead, you should be able to exploit *all* the avenues of physical human-robot interaction to convey your intent.

As robots increasingly share spaces with human partners, physical interaction between humans and robots becomes inevitable. Within this chapter we focus on robot arms that perform manipulation tasks. Here we can harness physical interaction as a channel for communication; when the human kinesthetically guides a robot arm throughout their desired task, the human is providing a *demonstration*; when the human pushes, pulls, or twists the robot to fix a part of its motion, the human is showing a *correction*; and when the human

physically observes the robot and ranks its performance, the human is indicating their *preference*. Demonstrations, corrections, and preferences all communicate information about how the human wants the robot to behave. But we recognize that these modalities provide different — and often complementary — types of information: demonstrations provide an outline of the high-level task, corrections hone-in on fine-grained aspects of the robot's motion, and preferences provide a ranked comparison between various repetitions of the same task. The right type of interaction (e.g., demonstration, correction, or preference) depends on the task, the user, and what the robot has learned so far.

Recent research on physical human-robot interaction develops separate approaches for each type of human feedback. Robot arms can learn manipulation tasks only from demonstrations [43], only from corrections [1], or only from preferences [2]. Moving beyond physical human-robot interaction, there is also work that unites combinations of these data sources: for instance, learning from demonstrations and preferences [4, 56, 57, 58, 59], or from demonstrations and corrections [15, 16, 54, 55]. But to learn from multiple sources of human feedback the robot must either (a) have prior information about the tasks the human has in mind [4, 62] or (b) apply reinforcement learning to identify the optimal trajectory [56, 57, 58, 59, 61]. These constraints present practical challenges during physical human-robot interaction: intuitively, we seek robots that learn new and unexpected tasks in real-time, without stopping to perform reinforcement learning through trial and error.

In this chapter we propose a formalism for learning from physical interaction that unites demonstrations, corrections, and preferences. We recognize that these different types of interactions provide different types of data about the human's desired task. To ground each feedback type within the same learning framework, our hypothesis is that:

We can unite physical demonstrations, corrections, and preferences under a single framework to learn a reward model that assigns higher scores to the human's input

trajectories as compared to any modified alternatives.

Applying this insight we develop a two-step algorithm. First, we observe the human's physical interactions with the robot and learn a *reward model* from scratch. Second, we exploit the underlying structure of manipulation tasks by applying *constrained optimization* to map the learned reward into a robot trajectory. This process is iterative and free-form: at every iteration the human can provide demonstrations, corrections, or preferences, and the robot updates its reward model and identifies an optimal trajectory in real-time. We emphasize that the reward model is a neural network that does not require any *a priori* knowledge about the human's potential rewards, and thus the current user is able to teach the robot arbitrary manipulation tasks. Returning to our running example, imagine that the robot has never assembled a chair before: but because the robot learns from the human's feedback to assign higher rewards to states where the chair legs are upright, the robot solves for a desired trajectory that carries the legs vertically.

3.2 Problem Statement

Going back to our motivating example from Figure 3.1, the human wants to teach their robot arm how to perform a manipulation task. To teach the robot the human exploits physical interaction: the human kinesthetically guides the robot through the process of attaching a chair leg, modifies specific sections of the robot's trajectory, and ranks the robot's autonomous behavior across repeated interactions. In this section we formulate the problem of learning from each of these different types of physical interaction. We explain how existing approaches combine demonstrations, corrections, and preferences when the robot has prior knowledge about the tasks it will perform, and then highlight the shortcomings of these assumptions. Finally, we define our proposed reward model: we highlight that this approach

can take advantage of the assumptions from previous works, but is also capable of learning new tasks that the robot does not know about beforehand.

Task. We formulate the task that the human wants the robot to perform as a Markov decision process: $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, T, r, H \rangle$. Here $s \in \mathcal{S}$ is the state of the robot and $a \in \mathcal{A}$ denotes the robot action. For instance, in our motivating example the robot's state s consists of its joint position and the position of objects in the scene, and the robot's action a is its joint velocity (e.g., a change in joint position). At timestep t the robot transitions to a new state based on the dynamics $s_{t+1} = T(s_t, a_t)$. The task ends after a total of H timesteps.

Remember that — although the human knows what task the robot should perform — the robot may be uncertain about the correct task. We capture this objective using the reward function $r : \mathcal{S} \rightarrow \mathbb{R}$. The reward function maps robot states to scalar values (where higher scores indicate better states). To complete the task successfully the robot must maximize the reward function; we will enable the robot to learn this reward function from physical interaction.

Trajectory. During each iteration of the task the robot moves through a sequence of H states. We refer to this sequence as a trajectory $\xi \in \Xi$ such that $\xi = (s_0, s_1, \ldots s_H)$.

Demonstrations. The first way that the human can physically communicate with the robot arm is by providing demonstrations (see Figure 3.2). During a demonstration the human kinesthetically backdrives the robot throughout a trajectory ξ_d. We refer to the set of demonstrations as $\mathcal{D} = \{\xi_{d_1}, \xi_{d_2}, \ldots\}$, where each element of $\mathcal{D}$ is an H-length trajectory that shows the entire task.

Corrections. During demonstrations the robot is passive throughout the entire interaction. But when the robot attempts to perform the task autonomously, the human may intervene to correct just a snippet of the robot's motion. Let $\xi_r \in \Xi$ be the trajectory that the

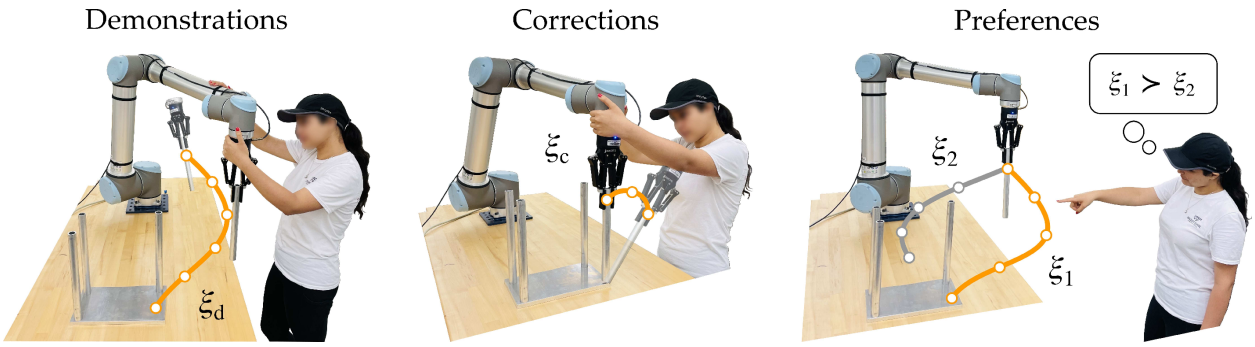


Figure 3.2: Different types of physical feedback. (Left) Humans can convey information to robot arms by kinesthetically guiding the robot through a demonstration of the task. Demonstrations provide high-level information about the entire trajectory. (Middle) To refine a specific part of the robot's motion humans may make physical corrections. These corrections fine-tune the robot's behavior. (Right) Over repeated interactions the human will observe multiple robot trajectories. Humans can rank these trajectories (i.e., give their preferences) to indicate when the robot is making a mistake. We note that preferences are not physical — in the sense that the human does not apply forces or torques — but preference feedback naturally emerges when humans and robots occupy the same space and the human can physically observe the robot's behavior.

robot is autonomously executing, and let $\xi_c \in \Xi_{\mathcal{C}} \subset \Xi$ be the human's correction. We emphasize that ξ_c includes fewer than H timesteps, and the human only physically intervenes to push, pull, and guide the robot during ξ_c. Let the set of human corrections be written as $\mathcal{C} = \{(\xi_{r_1}, \xi_{c_1}), (\xi_{r_2}, \xi_{c_2}), \ldots\}$. Each correction consists of both the robot's initial trajectory and the human's modification.

Preferences. For demonstrations and corrections the human is physically in contact with the robot arm. But the human and robot are also sharing the same space: and thus the human can observe the robot's trajectories and provide feedback about its behavior. Hence, the final way that the human conveys information to the robot is through preferences. We define a preference query as a set of k trajectories ranked in order by the human: $Q = \{\xi_1 \succ \xi_2 \succ, \ldots, \succ \xi_k\}$. Put another way, the human thinks ξ_1 better aligns with the task's reward than ξ_2. We group the human's preferences into a set $\mathcal{Q} = \{Q_1, Q_2, \ldots\}$, where each element of $\mathcal{Q}$ is a set of k ranked trajectories.

3.2.1 Preliminaries: Learning Rewards with Known Features

In order to formulate our problem we first need to explain how other methods learn rewards from demonstrations, corrections, and preferences. Remember that we defined the reward function r as an arbitrary mapping from states to scores. Related works such as [1, 4, 48, 62, 132] introduce structural bias by assuming that the reward function is a linear combination of *features*:

$$r_\theta(s) = \theta \cdot \phi(s) \tag{3.1}$$

Here $\phi : \mathcal{S} \to \mathbb{R}^n$ is the feature vector and $\theta \in \mathbb{R}^n$ is a weight vector. Features capture metrics that are potentially relevant to the current task. Within our running example from Figure 3.1, features could include the robot's distance from the chair, the angle of the chair leg, and the orientation of the chair. The robot is given these features *a priori* and must determine which features are important to the human; i.e., the robot is given ϕ and must learn θ. Sticking with our running example, the robot should learn to assign higher weight to the angle of the leg and lower weight to the orientation of the chair.

If we assume that the robot has access to the task-relevant features — and thus the reward is structured as in Equation (3.1) — we can use Bayesian inference to learn the correct weights θ. Let $P(\theta \mid \mathcal{D}, \mathcal{C}, \mathcal{Q})$ be the probability of weights θ given the human's previous demonstrations, corrections, and preferences. Applying Bayes' rule, and recognizing that each type of feedback is conditionally independent, we reach:

$$P(\theta \mid \mathcal{D}, \mathcal{C}, \mathcal{Q}) \propto P(\mathcal{D} \mid \theta) \cdot P(\mathcal{C} \mid \theta) \cdot P(\mathcal{Q} \mid \theta) \cdot P(\theta) \tag{3.2}$$

Here $P(\theta)$ is the prior distribution over θ, and the $P(\cdot \mid \theta)$ terms capture the likelihood of the observed demonstrations, corrections, or preferences given that the human has reward weights θ. Previous works have found expressions for these likelihood functions [62]. For

instance, we can model humans as approximately optimal, so that human inputs with higher rewards are increasingly likely [132]: $P(\xi_d \mid \theta) \propto \exp(\sum_{s \in \xi_d} \theta \cdot \phi(s))$. What is important here is that — if we assume access to the task-relevant features — inferring θ simplifies to Equation (3.2).

This preliminary approach makes sense if robots have access to all their reward features *a priori*. In practice, however, robots will inevitably face tasks they did not expect and features that were not pre-programmed [3]. Consider our motivating example: when we first bring this robot arm onto the factory floor, will the robot understand the features of chair orientation or leg attachments? Human users should not be forced to hand-engineer features for each new task and environment; when the features are available, the robot arm should make use of their structure — but when the human's physical interactions are not aligned with any known feature, the robot should not be constrained to misspecified feature spaces [6]. Instead, we will develop robots that can learn reward functions *without* requiring predefined features.

3.2.2 Problem: Learning Arbitrary Rewards from Physical Interaction

Our goal is (a) to learn the task reward function from demonstrations, corrections, and preferences and then (b) to leverage this learned reward to identify an optimal robot trajectory that performs the task autonomously. To remove the reliance on features — and enable the robot to learn arbitrary task rewards — we will model the reward function as a neural network:

$$R_\theta(\xi) = \sum_{s \in \xi} r_\theta(s) \tag{3.3}$$

Here θ is the weights of the neural network, $r_\theta(s)$ is the learned reward at state s, and $R_\theta(\xi)$ is the cumulative reward along trajectory ξ. If features are available we can incorporate them within this formulation. Define $s = \big(s, \phi(s)\big)$ as an augmented state vector which now includes both the system state and the features $\phi(s)$. Returning to Equation (3.3), we learn a reward model r_θ that maps this augmented state to reward values: cases where the task reward simplifies to $\theta \cdot \phi(s)$ are a special instance of our more general formulation.

We have chosen to learn a reward function because it provides an avenue to unify demonstrations, corrections, and preferences. In the next section we will develop an algorithm to train this reward function from each different type of physical interaction.

3.3 Unifying Demonstrations, Corrections, and Preferences

Our learning approach is based on comparisons. Recall that our underlying hypothesis is that the human's inputs — whether they are demonstrations, corrections, or preferences — are *intentional* improvements to the robot's behavior. Put another way, the reward model in Equation (3.3) should assign higher scores to human trajectories than to nearby alternatives. In this section we apply our insight to develop a unified learning algorithm. First, we explain how to generate trajectory deformations that we can compare against the human's inputs. Next, we train the reward function to score the human's demonstrations, corrections, or preferences higher than these noisy alternatives. Finally, we leverage constrained optimization to convert our learned reward model into a robot trajectory. Throughout this section we consider both passive communication (where the human chooses how to physically intervene) and active information gathering (where the robot prompts the human to

uncover the correct reward function). We emphasize that our resulting approach is flexible, and humans can teach the robot using whichever physical feedback modalities they prefer.

3.3.1 Learning the Reward Model

Generating Trajectories for Comparison. Given an input trajectory ξ we first seek to generate a nearby alternative $\hat{\xi}$. The intuition here is that the human chose to input ξ and not $\hat{\xi}$ — and thus the robot should assign higher rewards to ξ as compared to $\hat{\xi}$.

To create alternatives we propagate noisy perturbations along the input trajectory following the approach outlined in [133, 134]:

$$\hat{\xi} = \xi + M\lambda \tag{3.4}$$

Here $\lambda \in \mathbb{R}^H$ is a noise vector that the designer selects and $M \in \mathbb{R}^{H \times H}$ is a symmetric positive definite matrix that defines a norm on the trajectory space. Our approach is not tied to any specific choice of M or λ; however, for our experiments we selected the acceleration norm

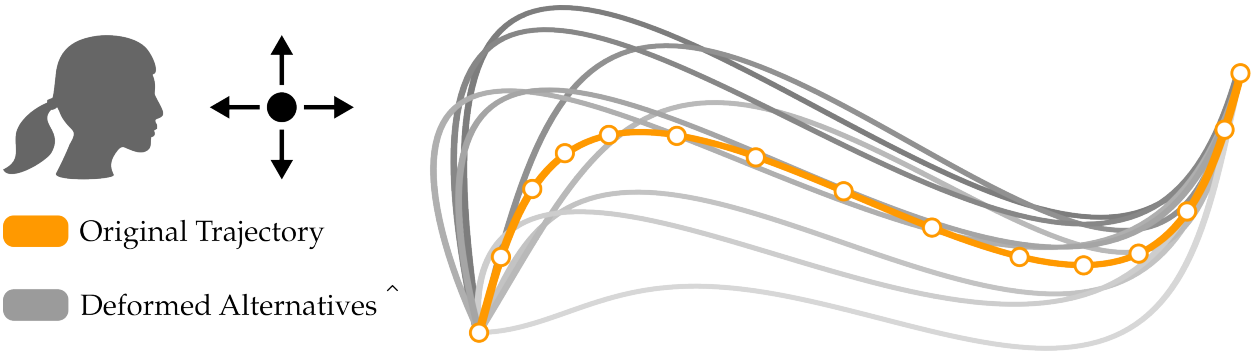


Figure 3.3: Generating trajectories for comparison. In this example the human moves a 2-DoF point mass robot along a sine wave. We record the initial trajectory ξ, and then apply Equation (3.4) to generate smooth perturbations $\hat{\xi}$. Our learned reward model should score ξ as a better trajectory than any of the alternatives $\hat{\xi}$.

[133, 134] in order to get *smooth* trajectory deformations of ξ:

$$M = (A^T A)^{-1}, \quad A = \begin{bmatrix} 1 & 0 & 0 & & 0 \\ -2 & 1 & 0 & & 0 \\ 1 & -2 & 1 & \cdots & 0 \\ 0 & 1 & -2 & & 0 \\ 0 & 0 & 1 & & 0 \\ & \vdots & & \ddots & \vdots \\ 0 & 0 & 0 & & 1 \\ 0 & 0 & 0 & \cdots & -2 \\ 0 & 0 & 0 & & 1 \end{bmatrix} \tag{3.5}$$

Each deformation uses the same M and a new vector λ. In our experiments we sampled λ from a zero-mean Gaussian distribution, and for every sampled λ we generated the corresponding $\hat{\xi}$ — to visualize this process, we show an example ξ and the generated alternatives $\hat{\xi}$ in Figure 3.3.

Learning from Demonstrations. Now that we have a way to generate nearby trajectories,

we will learn the reward function by comparing these alternatives to the human's inputs. We start with demonstrations. Assuming that the human provides near-optimal demonstrations, the robot should learn to match each demonstration $\xi_d \in \mathcal{D}$. More formally, the robot should learn a reward model such that $R\big(\xi_d\big) > R\big(\hat{\xi}_d\big)$, where $\hat{\xi}_d$ is a deformation found using Equation (3.4).

Let $P(x \succ y \mid \theta)$ be the likelihood — from the robot's perspective — that trajectory x has higher total reward than trajectory y given that the robot has learned reward weights θ. Inspired by prior work on human decision making and Luce's choice axiom [135, 136, 137], we write this probability as a softmax-normalized distribution:

$$P\big(\xi_d \succ \hat{\xi}_d \mid \theta\big) = \frac{\exp R_\theta\big(\xi_d\big)}{\exp R_\theta\big(\xi_d\big) + \exp R_\theta\big(\hat{\xi}_d\big)} \tag{3.6}$$

Remember that we want the robot to assign higher rewards to ξ_d as compared to $\hat{\xi}_d$. When this happens we have that $P\big(\xi_d \succ \hat{\xi}_d \mid \theta\big) \to 1$ in Equation (3.6). Accordingly, to drive the probability $P\big(\xi_d \succ \hat{\xi}_d \mid \theta\big) \to 1$, we train the reward model to minimize the cross entropy loss [3, 56, 57, 61]:

$$\mathcal{L}_{\mathcal{D}}(\theta) = -\sum_{\xi_d \in \mathcal{D}} \mathbb{E}_{\hat{\xi}_d \sim \xi_d}\Big[\log P\big(\xi_d \succ \hat{\xi}_d \mid \theta\big)\Big] \tag{3.7}$$

Note that in Equation (3.7) we sum the cross entropy loss across every demonstration $\xi_d \in \mathcal{D}$, and for each demonstration we sample a set of nearby trajectories $\hat{\xi}_d$. Reward models that minimize Equation (3.7) will assign higher scores to trajectories that are like the human's demonstrations, and lower scores to trajectories that are different from these demonstrations.

Learning from Corrections. During demonstrations the human backdrives the robot throughout the entire task; but during corrections, the human only fixes a snippet of the robot's trajectory. Given the robot's initial trajectory and the human's correction of this snippet — i.e., $(\xi_r, \xi_c) \in \mathcal{C}$ — we recognize that $R\big(\xi_c\big) > R\big(\hat{\xi}_r\big)$, where $\hat{\xi}_r$ is the segment of

the robot's trajectory that the human has intentionally modified. More generally, we assume that the human's correction shows the robot the right way to perform this part of the task. We therefore have $R(\xi_c) > R(\hat{\xi}_c)$, where $\hat{\xi}_c$ is a deformation of just the human's correction. Following the same derivation that we applied for demonstrations, we reach the following loss function for corrections:

$$\mathcal{L}_{\mathcal{C}}(\theta) = \sum_{(\xi_r, \xi_c) \in \mathcal{C}} -\log \frac{\exp R_\theta(\xi_c)}{\exp R_\theta(\xi_c) + \exp R_\theta(\hat{\xi}_r)} - \mathbb{E}_{\hat{\xi}_c \sim \xi_c} \left[\log \frac{\exp R_\theta(\xi_c)}{\exp R_\theta(\xi_c) + \exp R_\theta(\hat{\xi}_c)} \right] \quad (3.8)$$

Minimizing Equation (3.8) encourages the robot to learn a reward function that classifies ξ_c as a better trajectory than both the original segment $\hat{\xi}_r$ and perturbations $\hat{\xi}_c$ of the human's correction.

Learning from Preferences. As a final form of feedback the human operator can rank the robot's physical trajectories. Consider the robot arm in Figure 3.1: each time the robot attempts to add a chair leg, the human onlooker might score the robot's performance, and mark whether trajectory ξ_i is better or worse than trajectory ξ_j. We emphasize that these preferences provide a *direct* comparison between pairs of trajectories. Here we cannot assume that the human's preference is better than any other alternative; since the human's feedback only indicates that $\xi_i \succ \xi_j$, the reward model should learn $R(\xi_i) > R(\xi_j)$. Summing across the preferences $Q \in \mathcal{Q}$, we obtain the loss function:

$$\mathcal{L}_{\mathcal{Q}}(\theta) = -\sum_{Q \in \mathcal{Q}} \sum_{(\xi_i \succ \xi_j) \in Q} \log \frac{\exp R_\theta(\xi_i)}{\exp R_\theta(\xi_i) + \exp R_\theta(\xi_j)} \quad (3.9)$$

where $(\xi_i \succ \xi_j)$ is any pair of human-ranked trajectories from preference Q. Intuitively, Equation (3.9) trains the reward model to rank the robot's trajectories in the same order as the human operator.

The human may passively provide these rankings over the course of repeated interactions. Alternatively, the robot can *actively* prompt the human by rolling out a set of trajectories and asking for the human's preference. The goal of these active prompts is to reduce the robot's uncertainty about the correct task reward. To formalize this uncertainty we train an ensemble of m rewards with weights $\mathcal{E} = \{\theta_1, \theta_2, \ldots, \theta_m\}$. Each of these reward models is a separate instantiation of Equation (3.3), and is trained using the same demonstrations, corrections, and preferences. Let x and y be two robot trajectories. If all reward models agree on the relative scores of these trajectories — e.g., if $R_{\theta_i}(x) > R_{\theta_i}(y)$ for each $\theta_i \in \mathcal{E}$ — then the robot is confident that $x \succ y$. But in cases where the reward models disagree — for example, if $R_{\theta_1}(x) > R_{\theta_1}(y)$ while $R_{\theta_2}(x) < R_{\theta_2}(y)$ — then we do not know which trajectory is better at the task. At times when the reward models *disagree* the robot needs additional human feedback to resolve its uncertainty.

To actively learn about θ we will query the human by physically showing two robot trajectories, ξ_1 and ξ_2, and asking the human to choose which option they prefer. Humans may indicate that $\xi_1 \succ \xi_2$ or $\xi_2 \succ \xi_1$. Remember that we do not know beforehand how the human will respond to the robot; accordingly, we select ξ_1 and ξ_2 such that — no matter which option the human chooses — the robot maximizes the information it gains about the unknown task reward θ:

$$(\xi_1^*, \xi_2^*) = \arg\max_{\xi_1, \xi_2 \in \Xi} \mathcal{I}\big(\theta; \xi_i \mid (\xi_1, \xi_2)\big) \tag{3.10}$$

Here (ξ_1^*, ξ_2^*) is the greedily optimal query, ξ_i is the human's preferred trajectory, and $\mathcal{I}$ is the information gain [138]. Following the derivation in [4], we find that the expected information

gain from query (ξ_1, ξ_2) across the ensemble $\mathcal{E}$ of reward models becomes:

$$\mathcal{I}\big(\theta; \xi_i \mid (\xi_1, \xi_2)\big) = \frac{1}{m} \sum_{\xi_i \in Q} \sum_{\theta \in \mathcal{E}} P\big(\xi_i \succ \xi_{-i} \mid \theta\big) \log_2 \left(\frac{m \cdot P\big(\xi_i \succ \xi_{-i} \mid \theta\big)}{\sum_{\theta' \in \mathcal{E}} P\big(\xi_i \succ \xi_{-i} \mid \theta'\big)} \right) \tag{3.11}$$

where ξ_i is the trajectory the human prefers, ξ_{-i} is the other trajectory, and $P\big(\xi_i \succ \xi_{-i} \mid \theta\big)$ is the softmax-normalized distribution from Equation (3.6). In practice, Equation (3.11) looks for a query where (a) each reward model in the ensemble is confident about the relative scores of ξ_1 and ξ_2, but (b) some reward models think that $\xi_1 \succ \xi_2$, while other reward models think $\xi_2 \succ \xi_1$. We note that this active learning step is entirely optional. The robot still uses Equation (3.9) to learn from human preferences regardless of whether they are obtained passively or actively. However, as we will show in our experiments, actively selecting prompts using Equation (3.10) accelerates the robot's reward learning and resolves uncertainty across the ensemble of reward models.

Putting It All Together. We have identified loss functions that train the reward model to match the human's demonstrations, corrections, and preferences. For each of these interaction modalities we have a common theme: the human's inputs should be scored higher than the alternatives. Our final step is to unite Equation (3.7), Equation (3.8) and Equation (3.9) into a single loss function:

$$\mathcal{L}(\theta) = \mathcal{L}_{\mathcal{D}}(\theta) + \mathcal{L}_{\mathcal{C}}(\theta) + \mathcal{L}_{\mathcal{Q}}(\theta) \tag{3.12}$$

We note that Equation (3.12) is our parallel to the Bayesian inference from Equation (3.2).

We outline the implementation procedure for our approach in Algorithm 1. By controlling the number of comparisons for each feedback type (N_d, N_c and N_p), we can adjust their relative weight. Within our experiments we assign an equal importance to each of the

Algorithm 1 Learning from Multiple Forms of Feedback

Randomly initialize the ensemble of reward models with weights $\mathcal{E}^i = \{\theta_0^i, \theta_1^i, \cdots \theta_m^i\}$
Initialize the Demonstration, Correction and Preference buffers $\mathcal{D}, \mathcal{C}, \mathcal{Q}$
Initialize the number of noisy alternatives for $\mathcal{D}$, $\mathcal{C}$ and $\mathcal{Q}$: N_d, N_c, N_q
for $i = 0, 1, 2, \cdots$ **do**
    Initialize the rankings buffer $\mathcal{B}$
    if $i = 0$ **and** Demonstration Provided **then**
        $\mathcal{D} \leftarrow \xi_d$
    else if Correction Provided **then**
        $\mathcal{C} \leftarrow \xi_c$
    else if Preference Provided **then**
        $\mathcal{Q} \leftarrow Q$
    end if
    for $j = 1, 2, \cdots N_d$ **do**
        Generate noisy alternative $\hat{\xi}_d^j$ for $\xi_d \in \mathcal{D}$ ▷ see eq. 3.4
        $\mathcal{B} \leftarrow (\xi_d^j \succ \hat{\xi}_d)$
    end for
    for $j = 1, 2, \cdots N_c$ **do**
        Generate noisy alternatives $\hat{\xi}_c^j$ for $\xi_c \in \mathcal{C}$ ▷ see eq. 3.4
        $\mathcal{B} \leftarrow (\xi_c^j \succ \hat{\xi}_c)$
    end for
    for $j = 1, 2, \cdots N_q$ **do**
        Sample a preference $q^j \in \mathcal{Q}$
        $\mathcal{B} \leftarrow q^j$
    end for
    Update the reward models $\mathcal{E}^i$
end for

feedback types. For our approach, the users can choose to provide any form of feedback to the robot by indicating their choice on a user interface. Previous work has proved that it is optimal to start with passive forms of feedback (e.g. demonstrations) before collecting active feedback (see Theorem 2 in [4]). Following this, the default order for our simulations and user study is demonstrations, then corrections, followed by preferences.

Given the human's inputs, we train an ensemble of reward models that minimize $\mathcal{L}(\theta)$. Each reward model is a fully connected network with two hidden layers and leaky rectified linear activation units. The output of the reward model is bounded between -1 and $+1$ using

tanh(·). Our ensemble includes three independently trained reward models: each model optimizes its weights using the Adam learning rule with an initial learning rate of 0.001 [139]. To compute the reward of a state s we take the average score from $r_{\theta_1}(s)$, $r_{\theta_2}(s)$, and $r_{\theta_3}(s)$. We retrain the reward models after each new demonstration, correction, or preference from the human.

3.3.2 Optimizing for Robot Trajectories

The first half of our formalism is learning a reward function (or ensemble of reward functions) from the human's physical interactions. In the second part of our approach we convert this reward model r_θ into a robot trajectory ξ_r. Related approaches use reinforcement learning to identify the trajectory that maximizes r_θ [56, 57, 58, 59, 60, 61]; however, we recognize that reinforcement learning may not be appropriate for physical human-robot interaction. Here the human and robot are occupying the same space, and it becomes time consuming or unsafe for the robot to test multiple trajectories through the trial-and-error process of reinforcement learning.

Recall that our intended application is manipulation tasks for robot arms. Within this setting we take advantage of the underlying robot kinematics to solve for the optimal robot trajectory. More formally, we leverage constrained optimization to convert the reward model into a robot trajectory:

$$\xi_r = \arg\max_{\xi \in \Xi} \sum_{\theta \in \mathcal{E}} \sum_{s \in \xi} r_\theta(s) \quad \text{s.t. } \xi(0) = s_0,\ \xi(H) = s_H \tag{3.13}$$

Here s_0 is the start position of the robot arm (e.g., its current position) and s_H is a desired goal position. In practice, the goal position may not be known or there might not be a goal in the first place; in this case we leverage Equation (3.13) without the constraint

$\xi(H) = s_H$. Recent research on trajectory optimization has developed several approaches for Equation (3.13) [140, 141, 142]. Our formalism does not rely on any specific optimizer; in our experiments we use sequential quadratic programming to solve Equation (3.13) and identify the optimal robot trajectory ξ_r.

Summarizing our Algorithm. At the start of the i-th interaction the robot has an ensemble of reward models with weights $\mathcal{E} = \{\theta_1, \theta_2, \ldots, \theta_m\}$. The robot applies Equation (3.13) to identify the optimal trajectory under the learned rewards, and then uses shared control to track this desired trajectory ξ_r. The human onlooker may intervene to kinesthetically guide the robot through the task, physically correct the robot's motion, or rate the robot's overall behavior. We add this human feedback to the dataset of demonstrations $\mathcal{D}$ for the first interaction (i.e. if $i = 1$), and to the dataset corrections $\mathcal{C}$, or preferences $\mathcal{Q}$ for all other interactions ($i > 1$).The robot then updates its reward models to minimize the unified loss function in Equation (3.12) — the robot leverages these updated rewards to the start of interaction $i + 1$.

3.4 Simulation 1: Learning from Multiple Forms of Interaction

Now that we have developed a unified learning framework for physical human-robot interaction, we will compare different versions of our approach to the state-of-the-art baselines. As discussed in Section 2, several approaches learn from humans using physical interactions. Some of these approaches learn end-to-end models that capture the user's preferences for the task and learn a policy from the feedback provided, while others assume some knowledge over the features in the environment. In the latter, the features capture the task-specific

concepts and are assumed to be prior knowledge of the tasks that the robot may need to perform. In this section, we perform a detailed comparison of various physical interaction approaches that learn from demonstrations, corrections and preferences, and their various combinations.

Independent Variables. We test eleven algorithms that learn from physical interactions.

Among these eleven algorithms, seven are different versions of Our approach, i.e. using only demonstrations (**Ours (D)**), only corrections (**Ours (C)**), only preferences (**Ours (P)**), demonstrations + corrections (**Ours (DC)**), demonstrations + preferences (**Ours (DP)**), corrections + preferences (**Ours (CP)**) and demonstrations + corrections + preferences (**Ours (DCP)**). We include three baselines that learn end-to-end networks without using any predefined features – human-gated behavior cloning (**BC**) [141], adversarial inverse reinforcement learning (**AIRL**) [143] and a method for learning from demonstrations and preferences developed for Atari games (**Atari**) [56]. We also use one baseline that assumes prior knowledge of the features in the environment and learns from a combination of demonstrations, corrections and preferences (**RRIC**) [62].

During **BC**, the robot learns a policy from the human's initial demonstrations. The robot then shows the trajectory to the human and the human can intervene to physically correct and improve the robot's behavior at any point in the trajectory. **AIRL** utilizes the demonstrations and corrections provided by the human to recover a reward function. The robot then optimizes that reward function to generate new behaviors, and compares them to the human's original inputs. We used the repository from [59] to implement **AIRL**. **Atari** uses a two-step approach. First the robot leverages the human's demonstrations to learn a policy using imitation learning methods. The robot then shows the human sample trajectories generated using the learned policy, and the human indicates their preferences. These preferences are then used to learn a reward function that we optimize by applying the soft

actor-critic algorithm and generating queries in **Atari**. Finally, in **RRIC** the robot assumes full knowledge of the features in the environment and the reward function is modeled as a linear combination of these features. Based on the demonstrations, corrections and preferences provided by the human, feature weights are updated using Bayesian Inference. **Ours** directly learns a mapping from states to reward values and generates a trajectory to optimize that reward.

Procedure. The simulated human and a simulated robot performed two tasks (*Table* and *Laptop*) with each algorithm (see Fig. 3.4). For each of these tasks, the simulated human is teaching the robot to carry a cup of coffee to a goal position. In *Table*, the human wants the robot to carry the cup of coffee close to the table; in *Laptop*, the human wants the robot to avoid going over the laptop while moving to the goal location. For this experiment we used a 7-DoF Franka-Emika Panda robot arm. The robot *did not have access* to the task reward function. The simulated human knew their reward function and provided demonstrations, corrections, and preferences to optimize that reward and teach the robot.

Within this experiment we kept the number of interactions between the simulated human and the robot constant for each method. For **BC** and **AIRL**, the simulated human provided 6 demonstrations. For **Atari** the human first provided 2 demonstrations and then asked for 4 preferences to the human. **RRIC** received an even split for each feedback type: 2 demonstrations, 2 corrections and 2 preferences. Similarly, all different versions of **Ours** received an even split for each feedback type that version is meant to incorporate. For example, **Ours (D)** was given 6 demonstrations and **Ours (DC)** used 3 demonstrations and 3 corrections.

Dependent Variables. The simulated human worked with the robot to provide feedback across 6 interactions, and the robot optimized its learned reward to produce its final trajectory. We measured the performance of the robot by computing the *regret* of this learned

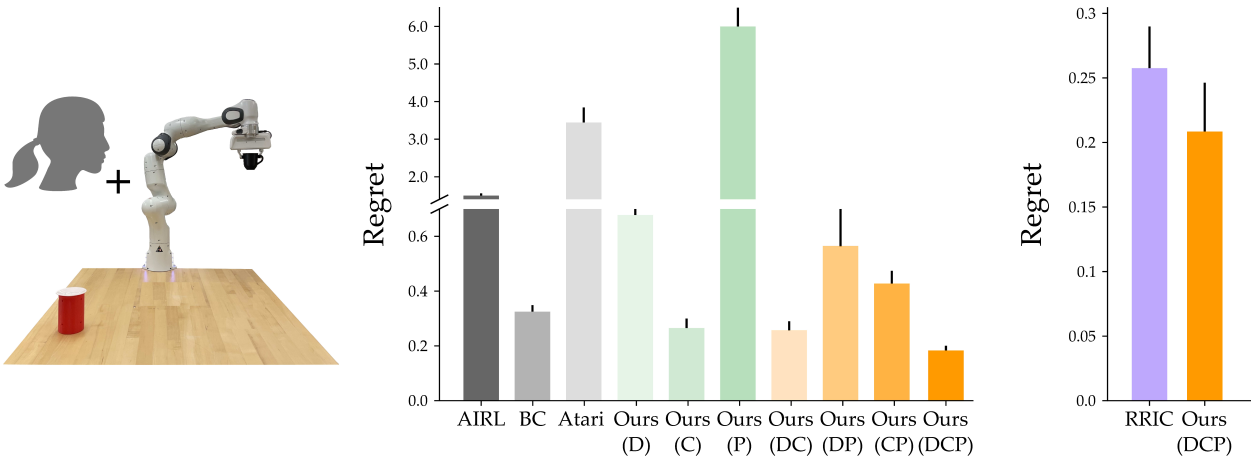


Figure 3.4: Experimental results for simulated humans paired with a Franka Emika robot arm. (Left) we compare different versions of our approach to state-of-the-art end-to-end learning baselines as well as a feature-based approach that combines multiple forms of feedback. (Center) 15 simulated humans perform each task (*Laptop* and *Table*) using all the end-to-end learning algorithms. (Right) The simulated humans perform each task with a feature-based learning algorithm. We record the performance of the robot after learning from each approach in the form of regret and report the average regret and standard error. **Ours (DCP)** significantly outperforms all other versions of our approach ($p < .05$). **Ours (DCP)** has a significantly lower average regret as compared to the end-to-end learning methods ($p < .05$) and performs at par with **RRIC**. We emphasize that **RRIC** has access to all relevant features in the environment, while **Ours** learns the reward function from scratch.

trajectory:

$$Regret(\xi) = \sum_{s \in \xi^*} r_{\theta^*}(s) - \sum_{s \in \xi} r_{\theta^*}(s) \tag{3.14}$$

where r_{θ^*} is the true reward function for the task, ξ^* is the optimal robot trajectory for the task, and ξ is the robot's learned trajectory. Regret quantifies how much worse the robot's learned behavior is than the ideal behavior: lower values of regret indicate better robot performance.

Hypothesis. For this simulation, we had the following hypotheses:

H1: *Our proposed approach with demonstrations, corrections, and preferences,* ***Ours (DCP)****, will outperform our approach with only one or two types of feedback.*

H2: ***Ours (DCP)*** *will outperform the end-to-end learning baselines.*

H3: *The performance of* ***Ours (DCP)*** *will match* ***RRIC*** *with known features.*

Results. Our results for this simulation are summarized in Fig. 3.4. Performing a repeated measures ANOVA test (Normality of data verified using Q-Q plots) with a Greenhouse-Geisser correction, we found that the robot's learning algorithm had a significant effect on the regret ($F(1.818, 140) = 61.939$, $p < 0.05$). By contrast, we found that the task did not have a significant effect on the regret ($F(1, 14) = 3.073$, $p = 0.101$). Thus, we report the combined regret for both the tasks in our results.

From the regret plots in Fig. 3.4 (Center), we observe that by combining all three forms of feedback our approach outperforms all other versions of **Ours** where we use only one or two forms of feedback ($p < 0.05$). This provides support for our hypothesis **H1**. We also observe that **Ours (C)** and **Ours (DC)** have a lower regret compared to **Ours (D)**, **Ours (P)**, **Ours (DP)** and **Ours(CP)**. This is a result of the iterative nature of the corrections as compared to the demonstrations, where the human provides all inputs at once before the robot updates its reward model.

We also notice that the end-to-end learning approaches perform at par with or better than some versions of **Ours** when only one or two types of feedback are available to our approach. While **Atari** and **Ours (DP)** receive the same forms of feedback, the regret for **Atari** is higher owing to the limited amount of data available for training. We observe that **BC** has a lower regret than **Ours (D)**, but performs at par with **Ours (C)**. This suggests that the performance of **Ours** improves when it receives incremental feedback from humans. However, when all three forms of feedback are made available to our approach, **Ours (DCP)** significantly outperforms all the end-to-end learning models ($p < 0.05$). This suggests that learning from multiple types of feedback is more effective than learning from just one or two types of feedback. Here, we find support for **H2**.

Finally, we compare **Ours (DCP)** to **RRIC**, which has access to the features in the environment and learns using all three forms of feedback within a Bayesian inference framework (see Fig. 3.4 (Right)). We observe that the performance of **Ours (DCP)** is comparable to **RRIC** ($p = 0.548$). This suggests that **Ours** — an approach that learns the reward end-to-end without any features — can perform as well as a Bayesian Inference approach that requires prior knowledge of all the features. Here we find support for our hypothesis **H3**.

3.5 User Study: Multiple Forms of Physical Interaction

So far we have evaluated our method in controlled experiments with simulated human users. In this section we will test our unified approach on *actual* participants. These participants physically interact with a 7-DoF robot arm by applying forces and torques, and communicate their intended task to the robot through demonstrations, corrections, and preferences. We compare our algorithm to interactive learning baselines that combine multiple types of human feedback. Here we explore scenarios where the robot must learn the task from scratch: our method and the baselines have no prior knowledge of the features or the tasks that the participants want to complete. Users must communicate their desired tasks through physical interaction. Videos of our user study are available at https://youtu.be/FSUJsTYvEKU.

Independent Variables. We tested four different algorithms that learn from physical interaction. Similar to Section 3.4, our baselines include human-gated behavior cloning (**BC**) [16], adversarial inverse reinforcement learning (**AIRL**) [143], and a method for learning from preferences and demonstrations developed for Atari games (**Atari**) [56]. **Ours** leverages the

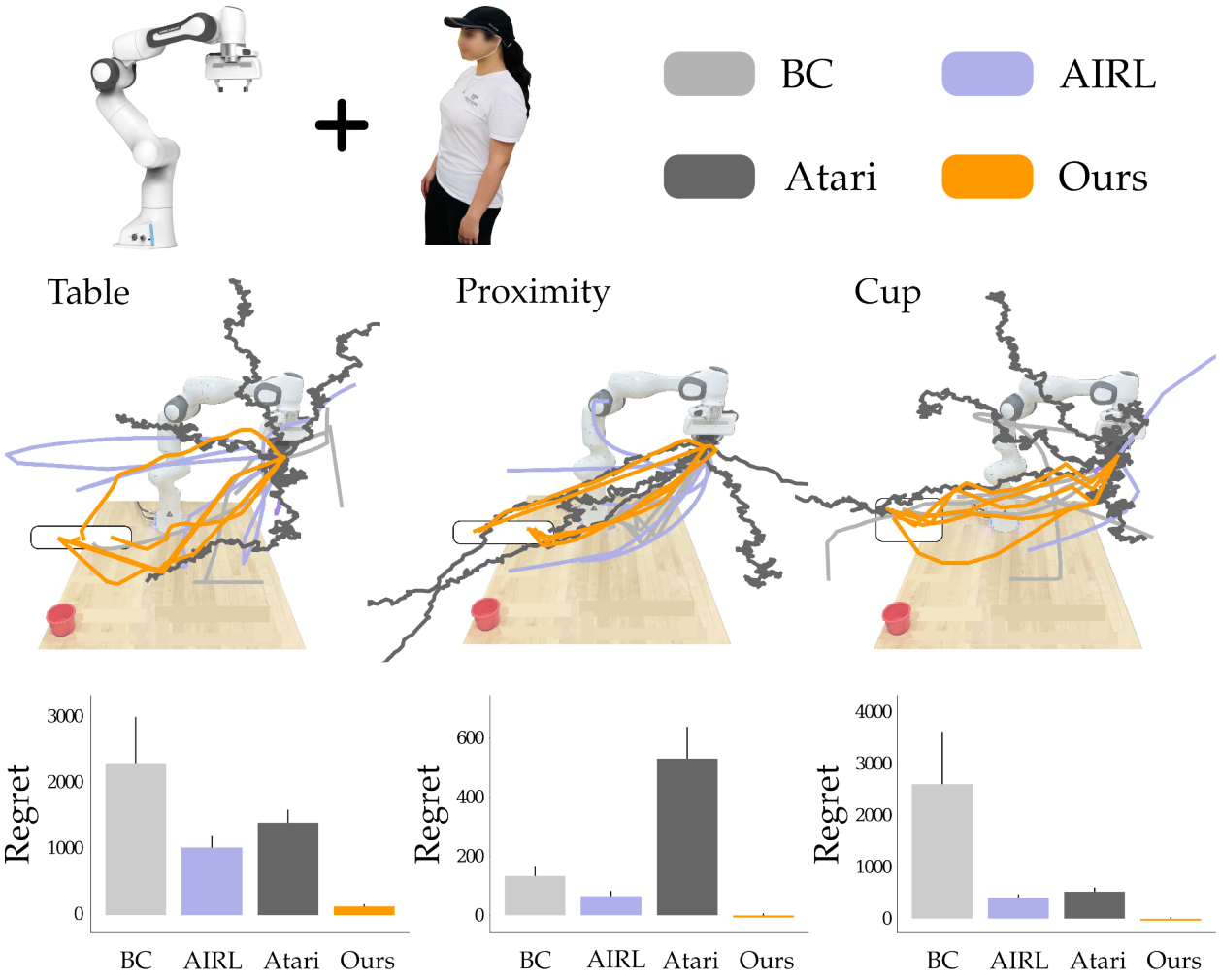


Figure 3.5: Learned trajectories and objective results from our in-person user study. (Top) Participants physically interacted with a 7-DoF robot arm that had no prior knowledge about the tasks. The robot learned from physical interactions using our approach and imitation learning baselines that combine multiple feedback modalities. (Middle) The final trajectories the robot learned with each method. Five users taught the robot the *Table* task, five users taught the *Proximity* task, and five users taught the *Cup* task. During each task the robot needed to reach a goal position within the white rectangle. We trace the xyz position of the robot's end-effector; within the *Cup* task the robot also needed to maintain specific orientations. (Bottom) The regret between the robot's learned trajectory and ideal trajectory. Lower values of regret indicate that the robot completed the task correctly, and the error bars plot standard error of the mean. **Ours** outperforms **AIRL** and **Atari** on the *Table* and *Cup* tasks ($p < .05$), and **Ours** has a lower regret than all the baselines for the *Proximity* task ($p < .05$).

unified algorithm introduced in Section 3.3. We emphasize that each of these approaches learns without pre-defined features. The implementation of each of these approaches follows the procedure described in Section 3.4.

Experimental Setup. Participants physically interacted with a 7-DoF Franka Emika robot arm. During the user study humans tried to teach this robot three tasks (see Figure 3.5). In *Table* the robot had to reach the goal while carrying a cup close to the table; in *Proximity* the robot had to move to a goal region while staying away from the user. Note that the nature of these two tasks is similar to that of the experiments performed in Section 3.4. In this user study we introduce a new task, *Cup*, where the participants teach the robot to complete a scooping action and then pour the cup at the goal position. We asked participants to mark their goal at the start of each interaction. To encourage more diverse human inputs, participants were instructed to change their goal position within a marked region between interactions.

Participants and Procedure. For our user study we recruited 15 participants from the campus community (5 female, average age 25 ± 4 years). Participants gave informed written consent prior to the start of the experiment under Virginia Tech IRB #22-308.

The participants were divided into three groups of five people. Each group of participants performed a single task (i.e., participants only taught the table task, the proximity task, or the cup task). Importantly, users taught this task with *all four* of the robot learning algorithms. The order of the algorithms was counterbalanced using a Latin square design: e.g., some participants began with **Ours**, and others began with **BC**. For each learning algorithm the human and robot started over from scratch: the robot had no prior information, and the human provided new demonstrations, corrections, or preferences to convey their task.

For **BC** and **AIRL** the human provided 6 demonstrations[1]. With **Atari** the participant first provided 2 demonstrations and then the robot asked for 4 preferences (to reach a total 6 interactions). Finally, observing the result from Section 3.4, that multiple forms of feedback enable a better learning, we provide our approach with all three forms of feedback. We divide **Ours** evenly between each type of physical feedback: users gave two demonstrations, corrections, and preferences (to maintain a total of $2 + 2 + 2 = 6$ interactions).

Dependent Variables. After participants finished providing their inputs, the robot leveraged its learning algorithm to identify a final trajectory. We measured how effectively this final trajectory completed the intended task. More specifically, we applied 3.14 to quantify the regret between the robot's actual behavior and the ideal task behavior.

We also administered a 7-point Likert scale survey [144] to assess the participants' subjective responses to each learning condition. Our survey questions were organized into four multi-item scales: how *easy* it was to physically provide feedback to the robot, whether the robot *learned* to perform the task correctly, how *flexible* the robot was to different types of physical interaction, and if they would *prefer* using this method in the future. Every participant completed this survey four times: once after they finished working with each robot learning algorithm.

Hypothesis. For our user study we had the following hypotheses:

H4: *Robots using our unified learning approach will perform the task better after the same number of physical human interactions.*

H5: *Participants will subjectively prefer our learning algorithm as compared to the baselines.*

Results. The objective results from our user study are presented in Figure 3.5, and the

[1]For **BC** and **AIRL** we gave users the option of providing corrections that only modify a segment of the robot's behavior. However, since the robot's learned behavior after two demonstrations was far from the user's intended task, participants chose to keep demonstrating the entire trajectory. Note that neither **BC** or **AIRL** can learn from preferences.

Table 3.1: Questions on our Likert scale survey. We grouped questions into four scales and tested their reliability using Cronbach's α. We explored whether providing feedback to the robot was easy, the robot learned the task, if the users liked the flexibility of using different feedback forms, and if they preferred to use the method in future. Computed p-values indicate if users preferred our approach to the baselines, where $*$ denotes statistical significance.

Questionnaire Item	Reliability	F(3,42)	p-value		
			BC	AIRL	Atari
—It was easy to provide feedback to the robot. —Providing feedback to the robot was challenging.	0.863	1.707	0.486	0.7	0.185
— The robot learned to perform the task correctly. — The robot's motion did not align with what I was trying to teach the robot.	0.911	11.982	p <0.05*	p <0.05*	p <0.05*
—I liked showing different types of feedback. —I preferred just showing one type of feedback repeatedly.	0.789	0.225	0.689	0.571	0.427
— I would use this method in the future. — I would prefer another approach that I tried if I was to do this again.	0.806	4.263	0.353	p <0.05*	p <0.05*

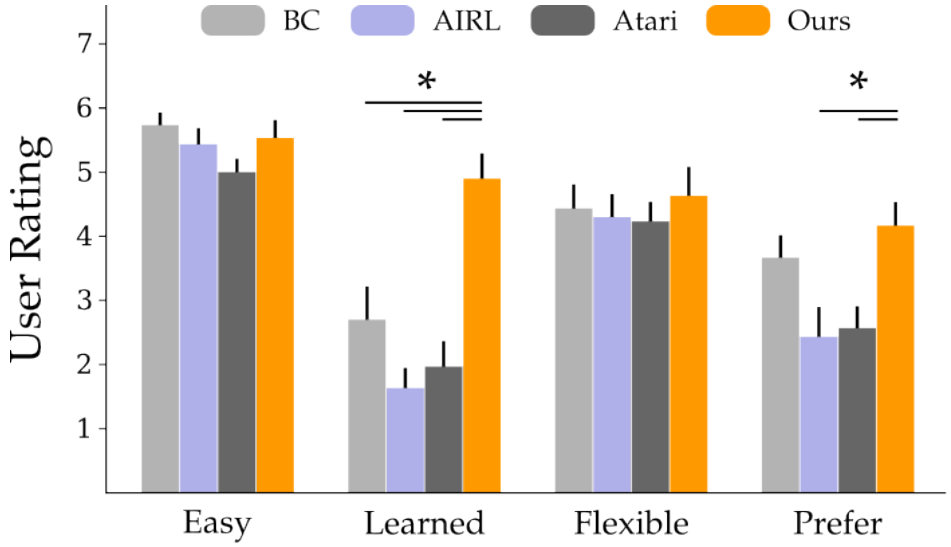


Figure 3.6: Subjective results from our in-person user study. Higher ratings indicate user agreement (e.g., a score of 7 indicates that it was *easier* to provide physical feedback). Error bars show standard error of the mean, and an $*$ denotes statistical significance ($p < .05$). After watching the final trajectory learned by each approach, participants rated **Ours** as a better *learner* than the baselines. Users also *preferred* **Ours** to **AIRL** and **Atari**.

subjective responses are summarized in Figure 3.6. Let us start with the objective results: After verifying the normality of the data using Q-Q plots and performing a repeated measures ANOVA test on our results, we found that the robot's learning algorithm had a significant main effect on regret ($F(3, 12) = 6.942$, $p < .05$). Looking at Figure 3.5 we notice that the regret for **Ours** is consistently lower than the baselines. Here lower regret is better — this indicates that the robot learned trajectories better matched the human's intended task. We can directly observe this trend from the final trajectories shown above: notice that **Ours** consistently moves to the goal region and completes the task, while the alternatives produce noisy, inconsistent motions. Post hoc comparisons confirm that **Ours** outperformed **AIRL** and **Atari** on the *Table* and *Cup* tasks ($p < .05$), and that **Ours** had an overall lower regret than all baselines for *Proximity* task ($p < .05$). We observe that given the limited amount of data (only 6 interactions), **Ours** has lower average regret with a low variance across all three tasks. On the other hand, **Atari** and **BC** fail to learn the task representations accurately from the same limited amount of data, and thus have a higher variance in their performance. Overall, the results shown here and in the video submission indicate that our unified learning approach was best able to synthesize the human's physical inputs and learn the correct task from scratch.

Next we consider the results from our Likert scale survey in Figure 3.6 and Table 3.1. After confirming that the scales were reliable (Cronbach's $\alpha > 0.7$), we grouped each scale into a single combined score and performed a one-way repeated measures ANOVA on the results. When users watched the robot's final learned behavior they perceived **Our** approach as a better learner than the baselines ($F(3, 42) = 11.982$, $p < .05$), and when they considered their experience teaching the robot they preferred to use **Ours** over the **AIRL** and **Atari** approaches ($F(3, 42) = 4.263$, $p < .05$). One confounding factor here is the about of time it took for the robot to learn from human's demonstrations. With **Ours** and **BC** the en-

tire process from teaching the robot to autonomously completing the task took roughly 10 minutes, while with **AIRL** and **Atari** it took more than 15 minutes on average (this additional time was needed to train the robot's policy). Participants may have preferred **Ours** and **BC** in part because they completed the training process more quickly. However, our overall results support hypothesis **H5** and suggest that participants subjectively perceived our unified approach as a better learner from demonstrations, corrections, and preferences.

3.6 Simulation 2: Learning with Known and Unknown Features

Now that we have tested our approach against baselines that learn end-to-end models, we will compare our approach to state-of-the-art baselines that use the knowledge about the features in the environment to learn the reward weights. As we discussed in Section 3.2.1, several related works learn from combinations of demonstrations, corrections, and preferences by assuming that the reward function is based on features. These features capture task-relevant concepts (e.g., the orientation of the chair leg) and are programmed using prior knowledge of the tasks the robot will need to perform. Here we consider situations in which the robot must learn *expected* tasks — i.e., cases where the features apply — as well as *unexpected* tasks where the pre-programmed features are insufficient. We conduct these experiments with simulated humans that provide inputs to real robot arms. Overall, we break this section down into two parts: (a) a comparison to physical interaction approaches that learn from demonstrations and corrections, and (b) a comparison to learning approaches that combine demonstrations and preferences.

3.6.1 Learning from Demonstrations and Corrections

Independent Variables. We consider two baselines for learning from physical demonstrations and corrections: **Coactive** [1, 2] and **FERL** [3]. In **Coactive** the robot assumes that it knows *all* the features for the current task; based on the human's inputs, the robot builds a reward function from these features and then selects the optimal trajectory. By contrast, in **FERL** the robot recognizes that it may be missing some task-related features. Here the robot leverages the feature demonstrations (feature traces) provided by the human to first learn the unknown features — once it has a model of these features, it then applies the same method as **Coactive** to build the reward function. Remember that our proposed approach (**Ours**) does not rely on features. Instead, we learn a mapping directly from states to rewards; for cases where the features are given, **Ours** can incorporate those features in the augmented state to be used as an input to our reward model (for learning and execution). Note that here, our reward model has the same information about the features as the other approaches (i.e. if **Coactive** and **FERL** have information about only *1 feature*, **Ours** also has information about only one feature).

Procedure. The simulated human and real robot performed three tasks with each learning algorithm (see Figure 3.7). For all three tasks the human is teaching the robot to reach a goal position. In *Table* the human wants the robot arm to move close to the table; in *Laptop* the human wants the robot arm to avoid passing above a laptop; and in *Cup* the human wants the robot arm to carry a cup upright so that it does not spill. Each task has two potential features: the goal that the robot should reach (e.g., distance to the goal) and the way the robot arm should move towards that goal (e.g., height from the table, distance from the laptop, or orientation of the cup). For these experiments we used a 6-DoF UR10 robot arm. This robot *did not know* the task reward function. To teach the real robot in a controlled setting, we used a simulated human: the simulated human knew the correct

reward, and provided demonstrations or corrections that optimized this reward.

We seek to understand how our approach compares to baselines both when the robot has prior knowledge about the task and when the task is new or unexpected. Accordingly, we tested three different conditions: (a) when the robot knows all task-related features, (b) when one task-related feature — *Laptop*, *Table* or *Cup* — is missing, and (c) when the robot does not know any features of the task.

Dependent Variables. The simulated human worked with the real robot to provide 20 inputs (i.e., demonstrations and corrections) over repeated interactions. For **Coactive** and **Ours** the first input is a task demonstration, and the remaining interactions are corrections. For **FERL** the type of input depends on the number of unknown features. When all the features are given, **FERL** also starts with a task demonstration followed by corrections. But when any feature is missing, the human first provides 10 feature demonstrations per missing feature. After these offline demonstrations (which are meant to teach features to the robot), the human provides one task demonstration and corrections similar to **Coactive** and **Ours**. Thus, each method receives one task demonstration and 19 corrections to learn the task (the feature demonstrations are not included as part of the 20 interactions). After each interaction we measured the performance of the learning robot using equation 3.14.

Hypothesis. For this experiment we had the following hypotheses:
H6: *Our method will match the baselines when all the features are known.*
H7: *Our method will outperform both* ***Coactive*** *and* ***FERL*** *when one or more features are missing.*

Results. Our results are visualized in Figure 3.7. Note that this figure is a grid: the columns are the tasks, and the rows are the amount of prior knowledge that the robot has about the tasks. To analyze the results we first verified the normality of our data using Q-Q plots and

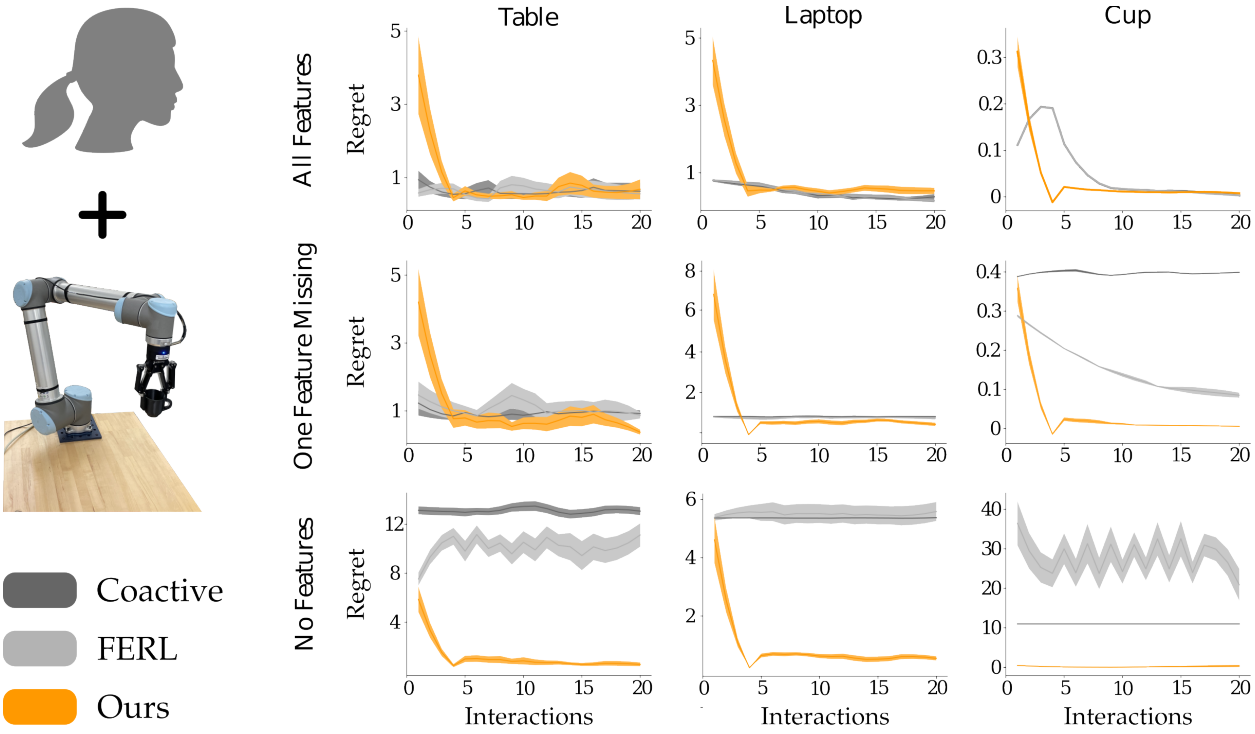


Figure 3.7: Experimental results for simulated humans paired with a UR10 robot arm. (Left) We compare our approach to two existing algorithms that learn from physical interaction. **Coactive** [1, 2] assumes that the reward is composed of pre-programmed features, and **FERL** [3] learns features from human demonstrations before constructing a reward function from those features. (Right) Over repeated interactions 10 simulated humans input demonstrations and corrections to teach the *Table*, *Laptop*, and *Cup* tasks. The columns correspond to the tasks, and the rows capture the prior information the robot has about each task. In the first row the robot is given all task-related features, in the middle row the robot is missing one feature, and in the bottom row the robot has no prior information about the task. The plots show regret (the difference in reward between the ideal trajectory and the learned trajectory), and the shaded regions show the standard error. At the end of all 20 interactions **Ours** performs similar to or worse than **Coactive** and **FERL** when all features are known. If one or more feature is missing, however, **Ours** outperforms both baselines.

then performed a repeated measures ANOVA with Greenhouse-Geisser correction. We found that the robot's learning algorithm had an effect on regret across all tasks and conditions: $F(1.040, 18) = 29.063$, $p < 0.05$.

To understand this result we next explored how the robot's performance changed based on the amount of prior information available to the learning algorithm. In the first row of Figure 3.7 we consider scenarios where all the task-related features are known. For example, during the *Table* task the robot knows that the reward is a function of the distance from the goal and the height of the end-effector. Overall, we found that — when all features are given — **Ours** performs on par with or worse than the baselines by the end of all 20 interactions. Post hoc tests revealed that for the *Table* task **Ours** matched the alternatives (**Coactive**: $p = 0.788$, **FERL**: $p = 0.790$). During *Laptop* our method performed similarly to **Coactive** ($p = 0.117$) but had higher regret than **FERL** ($p < 0.05$). Finally, within *Cup* both **FERL** and **Coactive** outperformed our approach at the last interaction ($p < 0.05$) These results partially support hypothesis **H6**: when the robot is given prior information about all the features, existing methods leverage this structure to accurately learn the human's reward. Our approach starts with worse performance because it does not make assumptions about the reward function and must learn to focus on the given features.

However, the relative performance changes once the robot encounters new or unexpected tasks. In the second row of Figure 3.7 we test settings where the robot knows one feature (distance to the goal) but does not know the other task-related feature. Because **Coactive** assumes that all the features are given, it treats each human input as an observation about the correct distance to the goal (and never realizes that the human's inputs may be communicating something else). **FERL** takes a step towards resolving this problem by trying to learn the missing feature from the first four interactions. But post hoc tests reveal that our proposed learning approach matches or outperforms the feature-based alternatives. For

the *Table* task there are no statistically significant differences between **Ours** and**Coactive** ($p = 0.074$), but **Ours** has a lower regret than **FERL** ($p < 0.05$). On the other hand, during *Laptop* and *Cup* our method leads to less regret than both the baselines by the end of the physical interactions($p < 0.05$). This trend continues in the final row of Figure 3.7 where the robot has no prior information about the task reward. Here **Ours** outperforms both baselines across all three tasks ($p < 0.05$). Overall, these results suggest that when the robot encounters situations where it has incomplete information, our unstructured reward learning approach is better able to capture the correct task than baselines that depend on task-related features. We therefore find support for **H7** when the robot is learning from physical demonstrations and corrections.

3.6.2 Learning from Demonstrations and Preferences

Independent Variables. So far we have compared our approach to baselines developed specifically for physical interaction when learning from demonstrations and corrections. Next, we turn our attention to alternate approaches that learn from demonstrations and preferences [56, 57, 58, 59, 62]: although these methods are not designed only for physical interaction, they can be applied to our setting. Here we compare **Ours** to **DemPref** [4], a recent approach that combines both demonstrations and preferences to build a model of the human's reward function. Like **Coactive** in the previous experiment, **DemPref** assumes that the reward function is composed of features, and the robot knows all the relevant features for the current task. In this experiment, we aim to study the trade-off between our proposed approach and approaches that utilize Bayesian inference to learn the task representation from human feedback.

The **DemPref** algorithm has two parts. First, the robot gets demonstrations from the

human to learn a rough estimate of the reward function; next, the robot actively queries the human to elicit their preferences and fine-tune the learned reward. Recall from Section 3.3 that under our proposed approach the robot can collect human preferences passively or actively. We will therefore consider two different versions of **Ours**: one where the robot gets the human's preferences from randomly chosen trajectories, **Ours (Passive)**, and one where the robot applies Equation (3.10) and Equation (3.11) to select preference queries that will maximize the information the robot gains about θ, **Ours (Active)**. To make the comparison as fair as possible, we have given both **DemPref** and **Ours** the same dataset of 1000 trajectories from which to choose their preference queries. Each query in this dataset was sampled by choosing a random goal in the robot's workspace followed by generating two noisy trajectories to the goal. Finally, to show that obtaining human preferences improves the performance of our approach, we also include **Ours (Demo)**, a baselines where the robot learns from only one demonstration (without ever considering the human's preferences).

Procedure. We perform this experiment in a controlled environment by pairing simulated humans with a 7-DoF Franka Emika robot arm (see Figure 3.8). The environment has three features: the distance of the robot from the bowl, the height of the robot from the table, and the distance between the robot and the ball. For each learning algorithm we simulated 500 humans with randomly selected reward functions that depend on these three features. Put another way, every simulated human assigns a different relative importance to the task features following Equation (3.1). The robot does not know the human's reward function *a priori*. Over repeated interactions, the robot attempts to identify the correct reward (and the corresponding optimal trajectory) from the demonstrations and preferences of the current human user. Each simulated user selects their inputs to noisily optimize their internal reward function.

This experiment is designed similarly to the simulation in Section 3.6.1. We want to explore

how our approach compares to **DemPref** in situations where the robot encounters a familiar task and settings where the robot is faced with new or unexpected tasks. We therefore compare **Ours** to **DemPref** (a) when all the features are known and (b) when one feature is missing. Recall that the task has three potential features. For case where one feature is missing, we performed separate trials where we removed either the first feature, the second feature, or the third feature; we then report the average across these runs. **Our** approach was never given any information about the features (i.e., **Ours** had no prior information about the task).

Dependent Variables. The simulated human worked with the real robot over 11 interactions. During the first interaction the human provides a demonstration, and during the next 10 interactions the robot collects preferences. After each interaction the robot solves for its best guess of the task trajectory: we measure the performance of the robot learner using *regret* as defined in 3.14.

Hypothesis. For this experiment we had the following hypotheses:

H8: *Our method will outperform **DemPref** when the robot does not know all the task-related features.*

H9: *Our method will converge to the correct trajectory more rapidly when choosing active preference queries as compared to passively collecting preferences or ignoring preferences altogether.*

Results. We summarize the results from this simulation in Figure 3.8. Remember that lower regret scores are better: After testing for the normality of data using Q-Q plots, a repeated measures ANOVA with Greenhouse-Geisser correction revealed that the learning algorithm had a significant main effect on regret by the end of the interactions ($F(1.994, 1996) = 366.897, p < .05$).

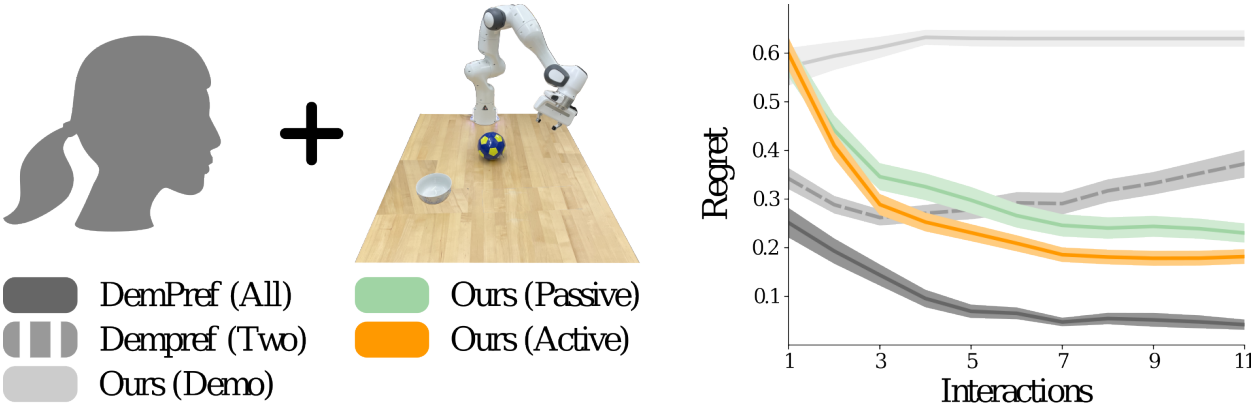


Figure 3.8: Experimental results for simulated humans paired with a Franka Emika robot arm. (Left) We compare our approach to **DemPref** [4], a method for learning from demonstrations and preferences that assumes the robot has access to all task-related features. (Right) 500 simulated humans attempt to teach the robot their desired task: each user provides one demonstration followed by 10 preferences. In **DemPref (All)** the robot knows all three task-related features, in **DemPref (Two)** the robot is missing one feature, and in **Ours (Demo)** the robot only observes a single demonstration. We compare these baselines to our approach when the robot chooses preference queries at random, **Ours (Passive)**, and when the robot asks questions to gain as much information as possible, **Ours (Active)**. The shaded region is the standard error. **Our** approach outperforms **DemPref** when a feature is missing and the robot must learn an unexpected task.

In settings where the robot encounters an expected task — i.e., when all the features of the environment are pre-programmed into the robot arm — **DemPref (All)** outperforms our proposed approach; **Ours (Passive)**: $p < .05$ and **Ours (Active)**: $p < .05$. However, when the robot is missing a task-related feature **DemPref (Two)** becomes confused by the human's feedback. Looking at Figure 3.8, we notice that **DemPref (Two)**'s performance decreases as the the human provides additional preferences: this occurs because the robot is misinterpreting the human's inputs as feedback about the two known features (instead of the one missing feature). By the end of the physical interactions our proposed approach better understands the unexpected task than the baseline; **Ours (Passive)**: $p < .05$ and **Ours (Active)**: $p < .05$. Overall, the results here agree with hypothesis **H8**. From this result, we conclude that when the robot has access to the environment features, Bayesian inference approaches perform on par with or better than our proposed reward learning approach. However, in more open-ended cases where information on reward functions is missing, **Ours** is more suitable for robot learning.

We next compared different versions of our learning algorithm. First, we find that learning from both demonstrations and preferences provides more information about the task than learning only from the human's initial demonstrations. Post hoc tests show that **Ours (Demo)** has significantly higher regret than **Ours (Passive)** ($p < .05$) and **Ours (Active)** ($p < .05$). Next, we tested to see whether actively selecting preference queries would lead to faster adaptation than passive human feedback. Remember that for **Ours (Active)** the robot asked questions using Equation (3.10), while for **Ours (Passive)** the robot chose preference queries at random. We observe that **Ours (Active)** has significantly better performance than **Ours (Passive)** across 500 users ($p < 0.05$). We conclude that hypothesis **H9** is supported when robots learn from demonstrations and preferences.

3.7 Conclusion

In this chapter we developed an alternate formalism for learning from physical human-robot interaction that unifies demonstrations, corrections, and preferences. When humans and robots share spaces, physical interaction is inevitable. Robots should leverage this interaction to learn from the human and improve their own behavior. But physical interaction takes many forms: humans can kinesthetically guide the robot throughout an entire task, fine-tune snippets of the robot's motion, or indicate which robot trajectories they prefer. Existing methods either learn from one of these interaction modalities, or combine multiple modalities by assuming prior information about the human's task. Instead, we introduce an end-to-end framework that (a) learns a reward function from scratch and then (b) optimizes this reward function to obtain robot trajectories.

Our key technical insight was that we can unite demonstrations, corrections, and preferences within the same framework by learning to assign higher rewards to these human inputs than to nearby alternatives. We first described a way to generate trajectory modifications. Next, we derived loss functions for learning from demonstrations, corrections, and preferences, and used these loss functions to train an ensemble of reward models. We also enabled robots to actively prompt the human and gain information about the correct task behavior. Finally, we converted the robot's learned rewards to robot trajectories using constrained optimization. Our framework was specifically developed for robot arms performing manipulation tasks: through simulations and a user study we compared our approach to multiple state-of-the-art baselines. Our results indicate that — when the robot knows what tasks to expect — our learning approach is comparable to existing methods that rely on pre-programmed features. However, when the robot encounters unexpected tasks (or when the robot must learn a new task from scratch), our method outperforms the interactive reward learning and imitation

learning baselines.

Limitations and Future Works. Our work is a step towards seamless communication between humans and robot arms. Because our system can learn new behaviors, one practical concern is *safety*: we must ensure that the robot learns trajectories that are safe for shared human-robot spaces. For instance, in our running example the robot arm should never learn to swing towards the human or run into the table. If the designer knows these safety constraints *a priori* we can embed them within our approach. Specifically, designers could augment Equation (3.13) to constrain the robot to have a certain workspace or speed thresholds. However, if designers do not impose any limits, we currently cannot guarantee that our robot will learn human-friendly behaviors.

One assumption throughout our work is that the human's inputs are noisily optimal. We assume that — when the human provides a feedback — on average their input is better aligned with their underlying reward function than the alternatives. However, in some settings and modalities the human's inputs may have a persistent bias, leading to suboptimal demonstrations, corrections, or preferences. Imagine a person teaching a robot to move across the table: if this human teacher cannot reach the opposite side of the table, all of their demonstrations may only move the robot part of the way to their intended goal. Existing works have explored methods for extrapolating from suboptimal demonstrations to reach performance that exceeds the given feedback [58, 145, 146]. We hypothesize that these methods could be combined with our reward learning approach by leveraging the diverse types of human feedback. For example, the user may have a persistent bias in their demonstrations but not their preferences (e.g., in our example the human cannot reach across the table but they can compare two trajectories that do so). Hence, we envision an iterative solution where the robot uses our approach to build a reward model from different types of feedback while identifying which of these modalities are biased and which are reliable.

Chapter 4

Stable and Robust Online Learning from Humans

In the previous chapter, we introduced an approach that can combine multiple forms of feedback when learning online from human. One assumption in that approach was that the human is noisily optimal and can always convey their desired feedback to the robot successfully. However, the human's actions can be biased, i.e. the human may not always be able to provide accurate feedback. If we have some knowledge of the tasks that the human may want to teach the robot, we can stabilize the robot's learning rule around these known tasks. In this chapter, we propose an approach that makes the robot's learning rule robust-by-design. This approach enables the robot to converge to the correct task representation under a larger set of human inputs.

4.1 Introduction

Modern robots can learn end-user preferences in real-time from human feedback. For instance, in Figure 4.1 a user physically corrects the robot arm to keep it away from a pitcher. Based on this human input, the robot should learn to consistently carry cups farther from pitchers. State-of-the-art paradigms for real-time learning apply online gradient descent, where the robot updates a point estimate over the human's preferences given the human's

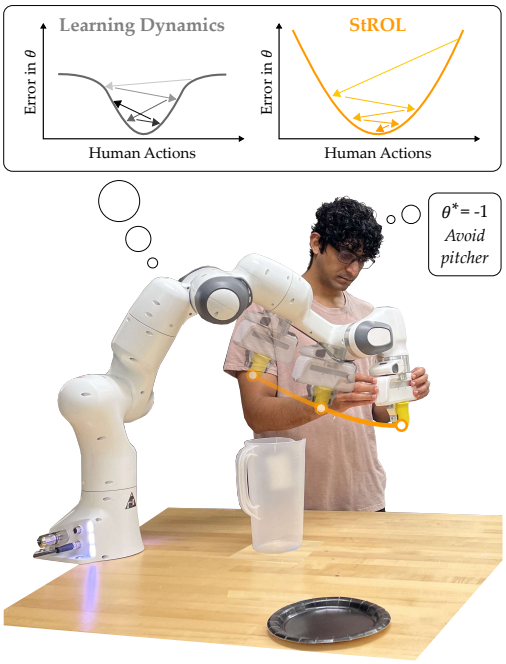


Figure 4.1: Human physically correcting a robot arm to convey their reward parameters θ^*. The robot learns online, and updates its point estimate θ after each human action. (Left) When the human takes noisy or suboptimal actions, the given learning dynamics can become unstable and fail to converge to θ^*. (Right) We learn how to modify these dynamics to expand the basins of attraction and increase robustness to imperfect human inputs.

feedback [2, 5, 6, 45, 47, 66, 67]. While this learning approach is effective if the user provides clear and unambiguous feedback (e.g., perfectly correcting the robot's motion), these approximate learning rules can be highly sensitive to noisy, biased, and suboptimal humans, leading to *unstable* robot learning [5]. In Figure 4.1, a human that over-corrects the arm causes the robot to oscillate between avoiding and approaching the pitcher, continually interacting without ever converging to the human's true preference. This raises the question, how can robots leverage online learning algorithms while ensuring robustness to suboptimal human feedback?

Instead of maintaining a fixed learning rule, and relying on the human's feedback to align with that learning rule:

We propose a control-theoretic approach that modifies the robot's learning rule to be more robust-by-design.

Specifically, we model the robot's learning algorithm as a *dynamical system* in the continuous space of preference parameters. This formulation enables us to apply Lyapunov analysis to robot algorithms that learn online from human inputs. We derive the basins of attraction, i.e., the range of human inputs that will cause the learning system to converge to the human's true preferences. We then introduce StROL, an algorithm that *modifies* the robot's learning rule to expand the basins of attraction, causing the robot's estimate to converge to the human's true preferences under a larger set of human inputs. Designers can leverage StROL to shape online learning rules to different users, tasks, and settings, enabling fast convergence despite suboptimal human feedback. Returning to Figure 4.1, with StROL the human can provide unintended forces — e.g., accidentally push too hard — and still convey their intended preference.

4.2 Problem Statement

We consider interactive scenarios where robots learn from humans in *real-time*. This includes settings where the robot performs a task and the human is purely a teacher (e.g., a human physically correcting a robot arm), or settings where the human and robot are both performing a task in the same environment (e.g., an autonomous car driving near a pedestrian). In both settings the human has a task that they want to perform, and the robot is trying to learn this task from the human's actions. Here we formulate real-time human-robot interaction as a dynamical system with two parts: *state dynamics* and robot's *learning dynamics*. We assume the state dynamics are known, and the robot is given some initial learning dynamics (i.e., the designer provides the robot with a baseline learning rule).

Physical Dynamics. Let $x \in \mathcal{X}$ denote the system state. In our experiments x can be the joint position of a robot arm, or the combined pose of an autonomous car and human pedestrian. At each timestep t the human takes action $u_{\mathcal{H}} \in \mathcal{U}_{\mathcal{H}}$ and the robot takes action $u_{\mathcal{R}} \in \mathcal{U}_{\mathcal{R}}$. The system state transitions according to the known *state dynamics*:

$$x^{t+1} = f(x^t, u^t_{\mathcal{H}}, u^t_{\mathcal{R}}) \tag{4.1}$$

The interaction ends after a total of T timesteps. We emphasize that the human and robot only collaborate for a *single* interaction; the robot *does not* repeatedly work with the same human across multiple, separate interactions.

Unknown Parameters. During interaction the robot optimizes its reward function. There may be some aspects of this reward that the robot already knows — e.g., the robot arm should carry water across the table. However, there are also parameters the robot does not know — like whether the robot should avoid moving over the pitcher. Let the true objective be $R(x, \theta^*) \rightarrow \mathbb{R}$, where $\theta^* \in \mathbb{R}^d$ is a d-dimensional vector of *correct* reward parameters (e.g., the task that the robot should optimize for). Returning to our motivating examples, θ^* could capture how the robot arm should carry a glass, or where and when the pedestrian will cross the road. The robot does not know θ^* and must learn these parameters from human data — specifically, observations of human actions.

Prior. Although the robot does not know θ^* a priori, we assume the robot is given a prior $P(\theta)$ over the continuous space of reward parameters. This prior captures which reward parameters θ are likely and unlikely. For instance, in Figure 4.1 the prior could be a bimodal distribution where it is likely that either (a) the human wants the robot to avoid the pitcher or (b) the human does not care about moving over the pitcher. In our experiments we hand-designed the priors as uniform or multi-modal distributions. More generally, these priors

could be gathered from human demonstrations, teleoperation data or pre-trained policies [147], obtained from data driven models of human state occupancy [148], or queried from large language models [75].

Learning Dynamics. The robot is trying to learn the true reward parameters θ^*. For tractable, real-time learning, the robot maintains a *point estimate* of these true reward parameters: this point estimate is the robot's best guess of θ^*. Let θ^t denote the robot's point estimate at timestep t, where $\theta \in \Theta$ lies in a continuous Euclidean space.

Building on the state-of-the-art in online learning from human feedback [2, 5, 6, 45, 67, 149], we use gradient ascent to capture the deterministic dynamics of the point estimate:

$$\theta^{t+1} = \theta^t + \alpha \cdot g(x^t, u_{\mathcal{H}}^t, u_{\mathcal{R}}^t, \theta^t) \tag{4.2}$$

Here $\alpha \geq 0$ is the learning rate and $g(x, u_{\mathcal{H}}, u_{\mathcal{R}}, \theta) \to \mathbb{R}^d$ is a function that determines how the point estimate changes in response to human action $u_{\mathcal{H}}$. We can think of Equation (4.2) as a dynamical system where θ is the "state" that updates at every timestep. We use the term *learning dynamics* to refer to Equation (4.2) and g interchangeably. The choice of g is up to the designer; in our analysis, the only requirement is that g in Equation (4.2) must depend on human action $u_{\mathcal{H}}$.

Example. Below we list one common choice of learning rule. Let $x_{\mathcal{H}} = f(x, u_{\mathcal{H}}, u_{\mathcal{R}})$ be the next state if the human takes action $u_{\mathcal{H}}$, and let $x_{\mathcal{R}} = f(x, 0, u_{\mathcal{R}})$ be the next state if only the robot acts. Related works [2, 5, 6, 45] update the point estimate to increase the reward for the human's corrected state $x_{\mathcal{H}}$ as compared to the default state $x_{\mathcal{R}}$:

$$g = \nabla_\theta \big(R(x_{\mathcal{H}}, \theta) - R(x_{\mathcal{R}}, \theta) \big) \tag{4.3}$$

We will use Equation (4.3) in our experiments. However, our underlying method is not tied to this specific instantiation.

Perturbations. We have formulated human-robot interaction as a dynamical system with state dynamics for x in Equation (4.1) and learning dynamics for θ in Equation (4.2). Ideally, the estimate θ should converge towards the human's preferences θ^* so that the robot learns the correct reward function. This could be straightforward if the human's inputs $u_{\mathcal{H}}$ exactly aligned with the robot's learning algorithm. Consider our motivating example of a human teaching a robot arm how to carry a cup: if the human physically corrects the robot such that $g(u_{\mathcal{H}})$ causes $\theta^{t+1} \rightarrow \theta^*$, then the robot will learn the correct task. But what if the human is not a perfect teacher? We recognize that humans are *suboptimal* agents [150, 151], and thus the dynamical system must be *robust* to perturbations in the human's actions.

4.3 Shaping the Learning Dynamics to Enlarge Basins of Attraction

In this section we present a control theoretic approach that modifies the learning dynamics to be more robust-by-design. Our proposed method is based on stabilizing the learning dynamics around the equilibrium $\theta = \theta^*$. More specifically, we leverage Lyapunov stability analysis in Section 4.3.1 to derive the condition under which the error between θ and θ^* is asymptotically decreasing. This condition defines the basins of attraction, i.e., the set of human inputs that cause the robot's point estimate θ to move towards the equilibrium θ^*. Next, in Section 4.3.2 we introduce StROL, an algorithm that modifies the learning dynamics to expand the basins of attraction. Given a prior over θ^* and/or a model of the human, StROL learns a correction term *offline* that is then added to the robot's original

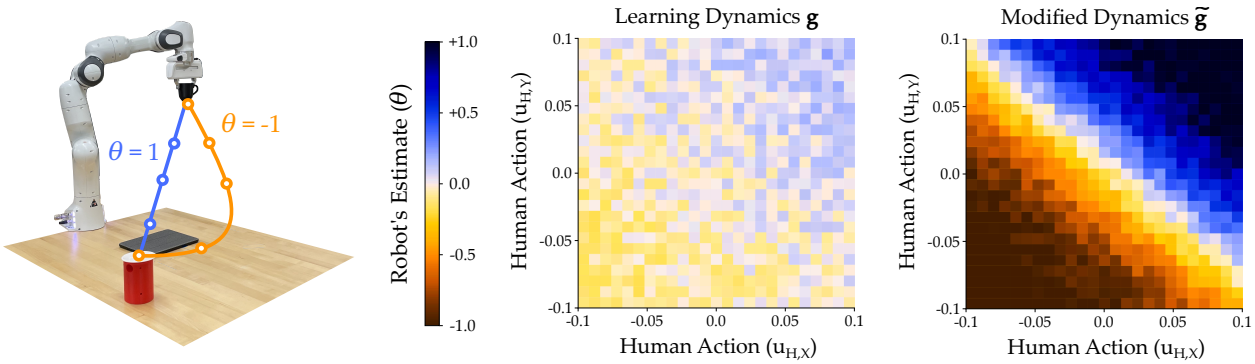


Figure 4.2: Example of how StROL autonomously generates robust-by-design learning rules that expand the basin of attraction. (Left) The robot does not know how it should carry a cup near a laptop. When $\theta = +1$ the human wants the robot to move straight to the goal, and when $\theta = -1$ the human wants the robot to avoid moving above the laptop. (Right) Plots of the robot's estimate θ as a function of the human's action $u_{\mathcal{H}}$ at the start state. With the original learning dynamics g the learning is inconsistent and gradual (i.e., nearby actions can convey either ignoring or avoiding the laptop). But StROL outputs the modified learning dynamics $\tilde{g}$ to expand the basin of attraction, so that nearby actions teach the robot the same parameters.

learning dynamics. We conclude with an example of our approach in Figure 4.2.

4.3.1 Deriving a Stability Condition

Humans do not always provide perfect, consistent inputs. Rather than assuming the human selects a single optimal choice of $u_{\mathcal{H}}$ to teach the robot, we are instead interested in the set of human actions that convey θ^*. Put another way, under what conditions does the human's action $u_{\mathcal{H}}$ cause the robot's estimate θ to converge to θ^*?

To answer this question, we first introduce the modified learning dynamics $\tilde{g}$. Under these new dynamics the robot's estimate θ updates according to:

$$\theta^{t+1} = \theta^t + \alpha \cdot \tilde{g}(x^t, u^t_{\mathcal{H}}, u^t_{\mathcal{R}}, \theta^t) \tag{4.4}$$

where Equation (4.4) matches Equation (4.2) with $\tilde{g}$ replacing the original term g. At this point we do not know what to choose for the robot's new learning dynamics. However, our choice of $\tilde{g}$ should cause the robot's estimate θ to converge towards the human's true reward parameters θ^*. Define $e^t = \theta^* - \theta^t$ as the error in the robot's estimate at the current timestep, such that the equilibrium occurs when $e = 0$ and $\theta = \theta^*$. To identify the set of human actions that cause the robot's estimate to converge towards this equilibrium, we will apply Lyapunov stability analysis.

Let the Lyapunov candidate function be $V^t = \|e^t\|_2^2$. Note that this function is positive definite and radially unbounded, i.e., the function cannot be 0 at any point except for the equilibrium ($\theta^t = \theta^*$) and $V^t \to \infty$ as $e_t \to \infty$. The time derivative of the candidate function is:

$$\dot{V} \cong V^{t+1} - V^t = \|e^{t+1}\|_2^2 - \|e^t\|_2^2 \tag{4.5}$$

For global asymptotic stability of the system around the equilibrium, according to Lyapunov's Direct Method we need that $\dot{V} < 0$ [152]. Substituting this condition into Equation (4.5), the sufficient condition for convergence becomes $\|e^{t+1}\|^2 < \|e^t\|^2$. Plugging in e^t and the modified learning dynamics from Equation (4.4), we reach:

$$\|\theta^* - \theta^t - \alpha \cdot \tilde{g}^t\|_2^2 < \|\theta^* - \theta^t\|_2^2 \tag{4.6}$$

Expanding this inequality and rearranging the terms, the sufficient condition for global asymptotic stability is:

$$\alpha^2 \|\tilde{g}^t\|_2^2 - 2\alpha(e^t \cdot \tilde{g}^t) < 0 \tag{4.7}$$

Equation (4.7) defines the basins of attraction as a function of the robot's new learning rule $\tilde{g}(x^t, u_{\mathcal{H}}^t, u_{\mathcal{R}}^t, \theta^t)$. Any human action $u_{\mathcal{H}}$ which satisfies Equation (4.7) will cause the robot's estimate θ to converge to the true parameters θ^*. These stable human actions lie within

the basins of attraction. Conversely, any human action $u_{\mathcal{H}}$ for which Equation (4.7) does not hold will cause the magnitude of the error to increase and drive θ away from θ^*. This set of unstable human actions lies outside the basins of attraction. We emphasize that the stability condition derived in Equation (4.7) depends on how $\tilde{g}$ maps the human's actions to changes in θ: a given human action may satisfy Equation (4.7) for one choice of learning dynamics $\tilde{g}$ but not for another. We also note that a more negative value in this constraint means that the human action $u_{\mathcal{H}}$ is causing θ to converge more rapidly.

4.3.2 StROL: Learning the Correction Term

Our Lyapunov analysis indicates that to enlarge the basins of attraction we need modified learning dynamics $\tilde{g}$ that satisfy Equation (4.7) across a wider range of human inputs. We instantiate these modified learning dynamics as the sum of the original term g and a *correction term* $\hat{g}$:

$$\tilde{g} = g + \lambda \hat{g} \tag{4.8}$$

where g is short for $g(x^t, u^t_{\mathcal{H}}, u^t_{\mathcal{R}}, \theta^t)$, $\hat{g}$ is short for $\hat{g}(x^t, u^t_{\mathcal{H}}, u^t_{\mathcal{R}}, \theta^t)$ and λ is a hyperparameter that regulates the relative weights of these terms. The designer provides the original learning rule g; our proposed StROL algorithm will autonomously generate the correction term $\hat{g}$. More specifically, $\hat{g}$ is a neural network that StROL learns offline so that the resulting $\tilde{g}$ satisfies Equation (4.7) for as many human inputs as possible.

In practice, if we are going to train $\hat{g}$ using Equation (4.7), we must be able to evaluate Equation (4.7) for different choices of $\tilde{g} = g + \hat{g}$. This means sampling true reward parameters θ^* (to get the error e) and sampling human actions $u_{\mathcal{H}}$ (to get g and $\hat{g}$). Under our proposed approach the robot samples these values from the prior over reward parameters and a nominal human model.

Prior. Within Section 4.2 we defined $P(\theta)$ as the prior over the human's reward parameters. Here we apply this prior to sample preferences $\theta^* \sim P(\cdot)$. Intuitively, by learning $\hat{g}$ across parameters sampled from $P(\theta)$, we are training the modified dynamics to more rapidly converge to reward parameters that are likely under the given prior.

Human Model. In addition to the prior, we assume access to a nominal human model. This human model inputs reward preferences θ^* and outputs actions $u_{\mathcal{H}}$ the human might take to teach those reward preferences. For example, an optimal human always takes actions $u^*_{\mathcal{H}}$ that align with the original learning dynamics and drive the robot's estimate $\theta^t \to \theta^*$. Offline, we can sample these optimal actions $u^*_{\mathcal{H}}$ by solving:

$$u_H^{*\,t} = \min_{u_H \in U} \theta^* - (\theta^t + \alpha g^t), \quad \theta^* \sim P(\theta) \tag{4.9}$$

In practice the human's actions are noisy and suboptimal. Without loss of generality, we write the actions of a suboptimal human as $u_{\mathcal{H}} = u^*_{\mathcal{H}} + \delta$, where δ represents the noise, bias or any other factor that perturbs the human. The choice of δ is up to the designer: StROL is not dependent on any specific human model. For example, in our experiments we set $\delta \sim \mathcal{N}(\epsilon, \sigma)$, where σ is the variance from the optimal actions and ϵ is a consistent bias.

One limitation of StROL is that it requires the designer to provide a prior and human model. Our experiments suggest that increasing the accuracy of both components will improve StROL's performance. However, neither component is strictly necessary: we find that StROL outperforms the baselines when given an accurate prior but inaccurate human model, and when given no prior but an accurate human model.

Offline Learning. We outline the offline training process for StROL in Algorithm 2. The robot first generates a synthetic dataset $\mathcal{D}$ containing reward parameters θ^* and human actions $u_{\mathcal{H}}$. This dataset is generated using the prior and nominal human model. Next, the

Algorithm 2 StROL: Stabilized and Robust Online Learning

Define original learning dynamics g ▷ see Equation (4.3)
Randomly initialize correction term $\hat{g}$
for $i = 1, 2, \cdots$ **do**
    Initialize the empty training dataset $\mathcal{D}$
    for $j = 1, 2, \cdots, N$ **do**
        Sample (x, θ, θ^*) tuple, where $\theta^* \sim P(\theta)$
        Get optimal actions $u^*_{\mathcal{H}}$ using Equation (4.9)
        $u_{\mathcal{H}} \leftarrow u^*_{\mathcal{H}} + \delta$
        Update the training dataset $\mathcal{D} \leftarrow (x, u_{\mathcal{H}}, \theta^*, \theta)$
    end for
    Compute the loss $\mathcal{L}$ using Equation (4.10)
    Update $\hat{g}$ to minimize $\mathcal{L}$
end for

robot evaluates the stability condition in Equation (4.7) across the synthetic dataset:

$$\mathcal{L} = \sum_{\theta^*, u_{\mathcal{H}} \in \mathcal{D}} \alpha^2 \|\tilde{g}^t\|_2^2 - 2\alpha(e^t \cdot \tilde{g}^t) \tag{4.10}$$

where loss function $\mathcal{L}$ is formed from the left side of Equation (4.7). The neural network $\hat{g}$ is then trained to minimize this loss function. Minimizing $\mathcal{L}$ optimizes the correction term $\hat{g}$ so that as many human actions from the dataset as possible lie within the basins of attraction and cause the robot's estimate θ to converge to the true parameters θ^*. As a result, StROL outputs new online learning dynamics $\tilde{g} = g + \hat{g}$ which are autonomously designed to be robust to suboptimal human inputs and enlarge the basins of attraction.

Example. In our experiments $\hat{g}$ is a fully connected 5 layer multi-layer perception with a rectified linear unit activation function. The output of $\hat{g}$ is bounded by a $\tanh(\cdot)$ activation function such that $\|\hat{g}\| \leq \|g\|$. This prevents the correction term $\hat{g}$ from overpowering the original learning dynamics g. In Figure 4.2 we show an example of how our corrective term modifies the learning dynamics to expand the basins of attraction. We first trained $\hat{g}$ offline using StROL (Algorithm 2). We next measured the estimate θ that the robot learned with

the original learning dynamics g and the modified learning dynamics $\tilde{g} = g + \hat{g}$. In this example $\hat{g}$ expands the basin of attraction so that one region of human actions teaches the robot to avoid the laptop ($\theta \to -1$), and the opposite region of human actions causes the robot to ignore the laptop ($\theta \to +1$).

4.4 Simulations

In Section 4.3 we presented StROL, and approach for learning robust-by-design learning dynamics. In this section we perform controlled simulations to examine how StROL compares to state-of-the-art baselines. We consider two simulated environments: (a) a multi-agent driving scenario where the robot car needs to learn the human's driving style to avoid a collision, and (b) a household setting where the human physically corrects a robot arm. In both environments we simulate suboptimal humans whose actions are sampled with increasing levels of noise and bias. We also perform simulations to evaluate the sensitivity of StROL to the hyperparameter λ and to test the performance of StROL when simulated humans change their reward preferences midway through the task (see A. For implementation details, see our repository here: https://github.com/VT-Collab/StROL_RAL.

Independent Variables. We compare our proposed algorithm (**StROL**) to four baselines that update θ using gradient-based learning rules. Gradient descent (**Gradient**) directly uses Equation (4.2) with learning dynamics g. Users who provide clear and unambiguous feedback can coordinate with **Gradient** to convey their reward preferences. But for suboptimal users, the robot's learning may be unstable and learn the wrong parameters. One-at-a-time (**One**) [5] modifies these learning dynamics to account for noisy and imprecise humans: instead of updating each element of θ at every timestep, the robot only changes the element of θ that best aligns with the human's action. The advantage of this approach is that it can help

filter suboptimal human inputs. However, one downside is that the robot only ever learns one reward parameter at a time, slowing down the overall learning. Misspecified Objective Functions (**MOF**) [6] also modifies the learning dynamics in Equation (4.2) to accommodate unexpected human behaviors. Specifically, here the robot ignores — and does not learn from — human actions $u_{\mathcal{H}}$ that are not aligned with any of the parameters in θ. Similar to **One**, **MOF** helps the robot filter out accidental and suboptimal human inputs. However, because the robot only learns from inputs that are optimal or close to optimal, this approach can cause the robot not to learn anything (i.e., keep θ constant) when interacting with very noisy humans.

Finally, we test an ablation of our proposed approach that we refer to as End-to-End (**e2e**). In **StROL** the robot's learning dynamics $\tilde{g}$ are the sum of the original dynamics g and the corrective term $\hat{g}$. We hypothesize that g provides an important starting point (i.e., the designer's knowledge) about the correct learning dynamics. In **e2e** we test whether including g is really necessary by setting $\tilde{g} = \hat{g}$, and training the robot's learning rule completely from scratch. **e2e** uses the exact same network architecture for $\hat{g}$ as **StROL**.

Environments. We tested two settings: a multi-agent **Highway** environment and a collaborative **Robot** environment.

In **Highway** a robot car is driving in front of a human car on a two-lane highway. We simulate both vehicles in CARLO [153]. The cars start at randomized positions in the left lane with the human behind the autonomous car. Both the human and robot cars have two-dimensional action spaces. For this simulation, we consider three features, (a) *distance* between the cars, (b) *speed* of the robot car and (c) heading direction of the human car indicating whether the human will *change lane*. The robot's goal is to minimize the distance travelled and avoid any collisions. To train the corrective term $\hat{g}$ in **StROL** and **e2e** we assume a bimodal prior: either (a) the human car will change lanes and then pass the robot

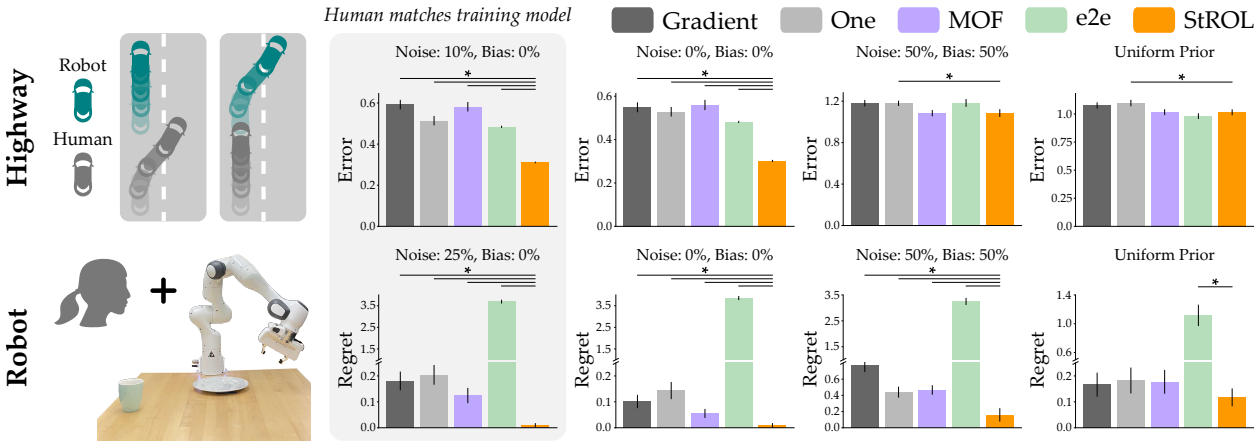


Figure 4.3: We compare StROL to state-of-the-art baselines in a multi-agent **Highway** environment (Top) and a collaborative **Robot** setting (Bottom). In **Highway**, the robot car takes turns interacting with 250 simulated human cars and tries to predict whether it should change lanes. We measure the *Error* between the robot's learned estimate θ and the simulated human's objective θ^*. In **Robot**, 100 simulated humans teach a 7 DoF Franka-Emika robot arm to reach for or avoid two stationary objects (also see Figure 4.2). We measure the *Regret* over the robot's learned behavior. For both environments we simulate humans with different levels of noise and bias. During offline training, **e2e** and **StROL** expected 10% noise in **Highway** and **25%** noise in **Robot**. The left column corresponds to this training setting. The other columns compare each method as the simulated human's noise, bias, and prior over θ^* deviates from the training data. An $*$ represents statistical significance ($p < 0.05$). A tabular version of these results is presented in our GitHub repository.

car (i.e. the human car does not care about *distance* but has a preference for speed and change lane), or (b) the human will follow the robot until the robot switches lanes (the human car does not want to *change lane* and maintains a minimum *distance* with the robot car). Both the agents choose their actions using a model predictive controller.

In **Robot** a simulated human corrects a collaborative robot arm. The robot's action space is its 3-DoF linear end-effector velocity. The environment includes two objects: a cup and a plate. The robot is not sure whether it should reach or avoid each object, and learns the human's preferences θ based on the human's corrections. When training the corrective term $\hat{g}$, the robot is randomly initialized in the environment and we assume that the human has

a bimodal prior over the features. The human likely prefers to either (a) reach the plate and avoid the cup or (b) go to the cup and avoid the plate. During each interaction the simulated human corrects the robot's behavior over the first 5 timesteps. After each timestep the robot updates its preferences θ and recomputes its trajectory to optimize for the learned reward function.

For both simulation environments we set the robot's initial estimate θ^0 as the mean over the prior $P(\theta)$. We also provided **Gradient**, **One**, and **MOF** with all the features of the task as a part of their learning rule g from Equation (4.3).

Dependent Variables. We measured the accuracy of the robot's learned estimate θ in both environments. In **Highway** we recorded the *Error* between the learned parameters θ and the true parameters θ^*, where $Error = \|\theta^* - \theta\|$. In the competitive, multi-agent highway environment, error is especially important because if the robot incorrectly estimates θ, the actions taken by the robot car can lead to a collision.

In the collaborative **Robot** setting, we explore whether the robot's learned behavior aligns with the human's preferences. We measure the *Regret* across the robot's learned trajectory:

$$Regret(\xi) = \sum_{x \in \xi^*} R(x, \theta^*) - \sum_{x \in \xi_\theta} R(x, \theta^*) \tag{4.11}$$

Here ξ^* is the optimal trajectory for reward weights θ^* and ξ_θ is the robot's learned trajectory (i.e., the trajectory that optimizes reward parameters θ. Regret quantifies how much worse the robot's trajectory is compared to the human's ideal trajectory: lower values indicate better performance.

Simulated Humans. We simulated humans with different priors and increasing levels of

suboptimality. More specifically, our simulated human chose actions according to:

$$u_h = u^*_{\mathcal{H}} + \delta, \quad \delta \sim \mathcal{N}(\epsilon, \sigma), \quad \theta^* \sim P(\theta) \tag{4.12}$$

where σ is controls the *Noise* and ϵ is the *Bias*. When training **StROL** and **e2e** we assumed a given level of noise and zero bias. When training in the **Highway** environment we set $\sigma = 10\%$ of the magnitude of the largest action, and in *Robot* we set $\sigma = 25\%$ of the magnitude of the largest action. For **Highway**, the the correction term $\hat{g}$ was trained offline for 1000 Epochs (i.e. generating dataset $\mathcal{D}$ and updating $\hat{g}$ 1000 times), while for **Robot**, $\hat{g}$ was trained offline for 500 Epochs. We then performed online simulations with increasing levels of noise and bias, and with changing priors $P(\theta)$. Hence, the simulated human's behavior *deviated* from the training behavior that our approach expected.

Hypothesis. We had the following two hypotheses:

H1. ***StROL*** *will outperform the baselines when the human's behavior is similar to the training behavior.*

H2. *When humans act in unexpected ways,* ***StROL*** *will perform better than or comparable to the baselines.*

Results. Our results are summarized in Figure 4.3. First we will breakdown these results for the **Highway** environment. Across all trials and conditions, a repeated measures ANOVA found that the robot's learning algorithm had a significant effect on learning error ($F(4, 996) = 32.1$, $p < 0.05$). Looking at the error plots in Figure 4.3 (Row 1, Columns 2-3), when the human actions at test time are similar to the human actions during training, **StROL** significantly outperforms all the baselines ($p < 0.05$). As the noise and bias in the human's actions increase (Row 1, Column 4), each algorithm performs similarly: **StROL** is not significantly different from **Gradient** ($p = 0.051$), **MOF** ($p = 0.98$), or **e2e:** ($p = 0.80$).

The same trend occurs when the human's θ^* are sampled from an unexpected prior (Row 1, Column 5). Put together, these results suggest that — when the human driver behaves similar to the designer's given model — **StROL** leads to robust robots that accurately predict θ. In the worst case — where the human significantly deviates from prior and human model — **StROL** is on par with existing methods.

We found similar trends when analyzing the **Robot** results. A repeated measures ANOVA with a Greenhouse-Geisser correction ($\epsilon = 0.552$) revealed that the choice of learning algorithm had a significant effect on the regret ($F(2.2, 218.5) = 1287.1$, $p < 0.05$). The plots in Figure 4.3 (Row 2, Columns 2-3) show that the robot's regret is significantly lower when the robot uses **StROL** ($p < 0.05$). As the humans become increasingly random, the regret for **StROL** increases, but it is still lower than the baselines ($p < 0.05$). On the other hand, if StROL is trained with an incorrect prior, **StROL** performs on par with the baselines, with no significant differences (**Gradient** ($p = 0.40$), **One** ($p = 0.30$), and **MOF** ($p = 0.31$)).

Interestingly, we observed that the relative performance of **e2e** changed between **Highway** and **Robot**. This may have occurred because of the complexity of the learning rule **e2e** needed to recover. In Highway the autonomous car can estimate the human's θ based purely on how the human changes lanes. By contrast, in Robot the system needs to account for both the robot's position and the human's inputs to recover θ. This suggests that we can learn the learning rules from scratch in simple settings, but as the environment becomes more complex, incorporating the original learning dynamics g becomes increasingly important.

4.5 User Study

To evaluate our approach in real-world environments, we next conducted an in-person user study where participants interacted with a 7-DoF Franka-Emika Panda robot arm. During

each trial users attempted to teach the robot their desired reward by applying forces and torques to the robot arm. We compared StROL to state-of-the-art approaches that learn online from human interventions [5, 6]. Implementation details and videos of our user study are provided here: https://github.com/VT-Collab/StROL_RAL

Independent Variables. We trained **StROL** offline using Algorithm 2. Similar to the simulations in Sections 4.4, our baselines include **One** [5] and **MOF** [6].

Experimental Setup. A 7-Dof Franka-Emika robot arm carried a cup across a table that contained a plate and a pitcher of water (see Figure 4.1). The robot started each trial by following a trajectory generated using randomly initialized feature weights θ^0. Users then physically intervened to correct the motion of the robot arm to teach it three different tasks. For **Task 1** users taught the robot to carry the cup to the *plate*, while keeping the cup close to the *table* and away from the *pitcher*. **Task 2** was similar to **Task 1**, with the addition that the users had to teach the robot to carry the cup at the correct *orientation*. Finally, in **Task 3** the users taught the robot to move away from all objects while keeping the cup upright. Task 1 had three features ($\theta \in \mathbb{R}^3$) while Tasks 2 and 3 had four features ($\theta \in \mathbb{R}^4$). These manipulation tasks with physical human corrections were similar to the user study environments used in [6] and [5]. When training **StROL** offline the robot's multimodal prior included **Task 1** and **Task 2**, but **Task 3** involved a new region of reward parameters that the learner did not expect.

Participants and Procedure. We recruited 12 participants from the Virginia Tech community (6 female, average age 23.5 ± 3.08). Participants gave informed consent prior to the start of the experiment under Virginia Tech IRB $\#22 - 755$.

The participants performed all three tasks with each learning algorithm. The order of the learning algorithms was counterbalanced using a Latin square design (e.g., some participants

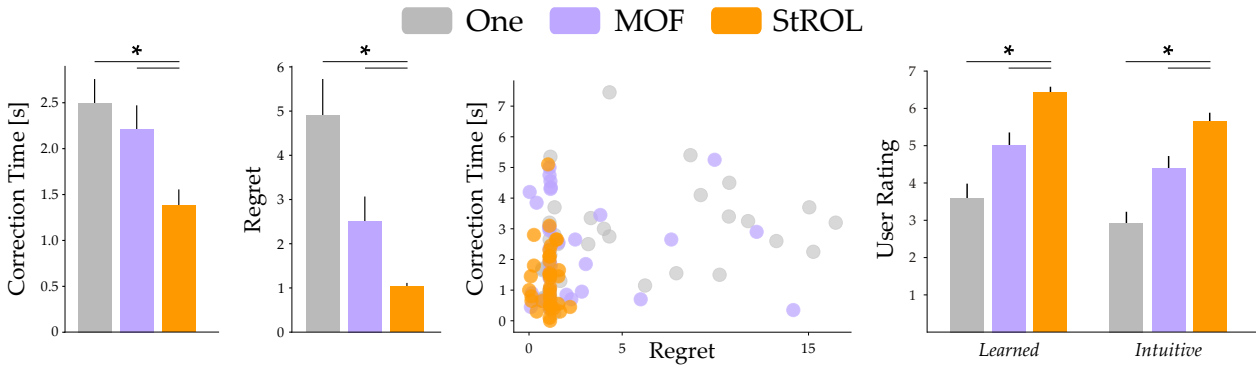


Figure 4.4: Objective and subjective results from the user study in Section 4.5. Participants physically interacted with a 7-DoF robot arm (see Figure 4.1) to teach it three different tasks. The robot used StROL or other online learning methods [5, 6] to infer the human's reward parameters in real-time. (Left) The time users spent correcting the robot and the regret across the robot's learned trajectory averaged over all three tasks. (Middle) For each individual task and participant (3 tasks $\times$ 12 participants) we plot their regret vs. correction time. (Right) The average user ratings from our 7-point Likert scale survey. Error bars show SEM and an $*$ denotes statistical significance ($p < 0.05$). A tabular version is presented in our GitHub repository.

started with **StROL**, others started with **One**, etc.). Before each task the robot played the ideal trajectory for that task (i.e., the robot showed the behavior that the participant should teach to the robot). Between each trial the robot reset from scratch: the robot did not carry over what it learned about θ from one trial to another.

We trained **StROL** offline to shape the learning dynamics. During training we used the noisy human model in Equation (4.12) with $\sigma = 25\%$ of action magnitude and $\epsilon = 0$. The multimodal prior $P(\theta)$ used during training consisted of 3-4 modes; these modes includes the desired behaviors for **Task 1** and **Task 2**, but not for **Task 3**. We emphasize that **StROL** was trained offline with simulated human data, and then deployed online to perform zero-shot learning with real humans and improve the overall robot performance.

Dependent Variables. To analyze how accurately the robot learned, we measured the robot's *Regret* according to Equation (4.11). To analyze how rapidly the robot learned, we measured the total amount of time the human spent correcting the robot arm (*Correction*

Time). We also administered a 7-point Likert scale survey to access the participants' subjective responses. Our survey questions were organized into two multi-item scales: whether the users thought the robot *learned* to perform the task correctly, and how *intuitive* it was for participants to teach the robot.

Hypothesis. We had the following hypotheses for this study:

H3. *With **StROL** users will teach the robot more quickly (shorter correction time) and accurately (lower regret).*

H4. *Participants will find **StROL** to be a more intuitive learner as compared to the baselines.*

Results. We first explore hypothesis **H3**, and refer to the objective results portrayed in Figure 4.4 (Column 1-3). A Repeated Measures ANOVA revealed that robot's learning algorithm had a significant effect on the correction time ($F(2, 22) = 5.602$, $p < 0.05$) and regret ($F(1.332, 14.651) = 9.108$, $p < 0.05$). Post hoc comparisons showed that **StROL** had significantly lower correction time and regret as compared to the baselines ($p < 0.05$) (see 4.4 Column 1-2). Column 3 in Figure 4.4 shows how a scatter plot of how the regret for each learning algorithm varied with the correction time. Across all participants and tasks, we observed consistently lower regret with **StROL**. But with **One** and **MOF**, there were some cases where the teacher spent a long time correcting, and the regret remained high. With **One** and **MOF** we also observed cases where the participants gave up teaching after a few corrections, leading to a short correction time and high regret.

To explore hypothesis **H4** we refer to the Likert scale survey in Figure 4.4 (Column 4). After verifying that the scales used for the survey were reliable (Cronbach's $\alpha > 0.7$), we grouped the responses for each scale into a combined score. A repeated measures ANOVA ($F(2, 70) = 21.301$, $p < 0.05$) suggested that the users perceived **StROL** to be significantly more *intuitive* than the baselines ($p < 0.05$). Similarly, a repeated measures ANOVA with a Huynh-Feldt correction ($\epsilon = 0.807$, $F(1.6, 56.5) = 18.1$, $p < 0.05$) revealed that after

observing the robot's final behavior, the users thought **StROL** *learned* better than the baselines ($p < 0.05$).

4.6 Conclusion

In this paper we presented a control-theoretic approach to learn robust-by-design online learning rules for human-robot interaction. We introduced StROL, an algorithm that modifies the robot's original learning dynamics to enlarge the basins of attraction and cause the robot's estimate θ to converge to the human's true preferences θ^* under a wider range of human actions. Our simulations and user study show that robots can apply the modified learning rules produced by StROL to more accurately and rapidly infer the preferences of noisy, suboptimal, and real-world users.

Limitations. Our proposed approach augmented the initial learning dynamics g with a correction term $\hat{g}$ to reach the modified learning rule $\tilde{g} = g + \hat{g}$. The relative weights of g and $\hat{g}$ must be tuned by the designer. If $\hat{g}$ is unbounded, the learned correction term may override g and constrain the robot learner into the basins of attraction, preventing the human from teaching reward parameters θ that lie outside of these basins. Conversely, if the designer constrains $\hat{g}$ to be too small, then StROL will not have a significant effect on the robot's learning. In general, we recommend using a smaller value for λ when the robot does not have access to a reliable prior or nominal human model. One possible way to tackle this limitation and automatically tune the relative weights of g and $\hat{g}$ could be inspired by [154]. During interaction the robot could infer how close-to-optimal the human teacher is using Bayesian inference. For near optimal humans the robot could increase the weight of g so that the human's teaching is not adjusted by StROL. Conversely, for increasingly suboptimal humans the robot could increase $\hat{g}$ and leverage the robust learning facilitated by StROL.

Chapter 5

Controlling Covariate Shift with Stable Behavior Cloning

So far we have explored scenario of online learning, where the human is available to provide feedback to the robot to correct and improve its behaviors. But, a human may not always be available to provide feedback. In such scenarios, the robot has to learn from the demonstration dataset that is provided. However, there may be states that the robot does not observe in this dataset but encounters during task execution. In such situations, the robot policy may take incorrect actions that may lead to unexpected outcomes. In this chapter, we propose a behavior cloning algorithm that leverages local stability analysis to encourage policy stability around the states seen in the demonstration dataset, without the need for additional data or data augmentation.

5.1 Introduction

Behavior cloning enables robots to learn new tasks by imitating humans. Consider the air hockey game in Figure 5.1. Here a human might show the robot arm a few examples of how to block the puck. With behavior cloning, the robot learns to match the human's actions, so that — if the robot sees the puck moving like it did in one of the examples — the robot mimics how the human expert blocked that puck. But what happens when the puck moves in

a new way (e.g., with a previously unseen angle or velocity)? This is an instance of *covariate shift*, a difference between what the robot observes at training time and what the robot encounters when executing its learned policy [20, 21, 155]. Covariate shift is a fundamental problem for behavior cloning because it can lead to *compounding errors*: a small change in the puck angle or velocity may cause the robot to take the wrong action, resulting in a larger covariate shift and increasingly incorrect robot behavior (i.e., the robot misses the puck entirely).

Existing research tries to prevent compounding errors and learn robust policies by focusing on the *data* (i.e., the human examples) used during behavior cloning. For instance, when the human provides initial demonstrations of their desired behavior, off-policy methods perturb the human to collect more diverse examples [79, 80], or synthetically augment the human's demonstrations to gather a larger dataset [22, 82, 83]. Similarly, as the robot executes what it has learned in the environment, on-policy methods encourage the human to correct the robot when it makes mistakes, and then add these new examples to the robot's dataset [16, 31, 55, 84, 85, 86]. In either case, a core idea within existing works is that — if the robot has sufficient and relevant data — it will return to the desired behavior when it encounters novel states and situations.

In this chapter we introduce an alternate viewpoint for robust behavior cloning. Instead of enhancing the data used to train the robot, we will focus on the control theoretic properties of the robot's learned policy. We hypothesize that:

Behavior cloned policies are robust when they stick to what they know, i.e., when the robot remains close to the example motions that the expert demonstrated.

Our intuition here is that compounding errors occur when the robot drifts away from its

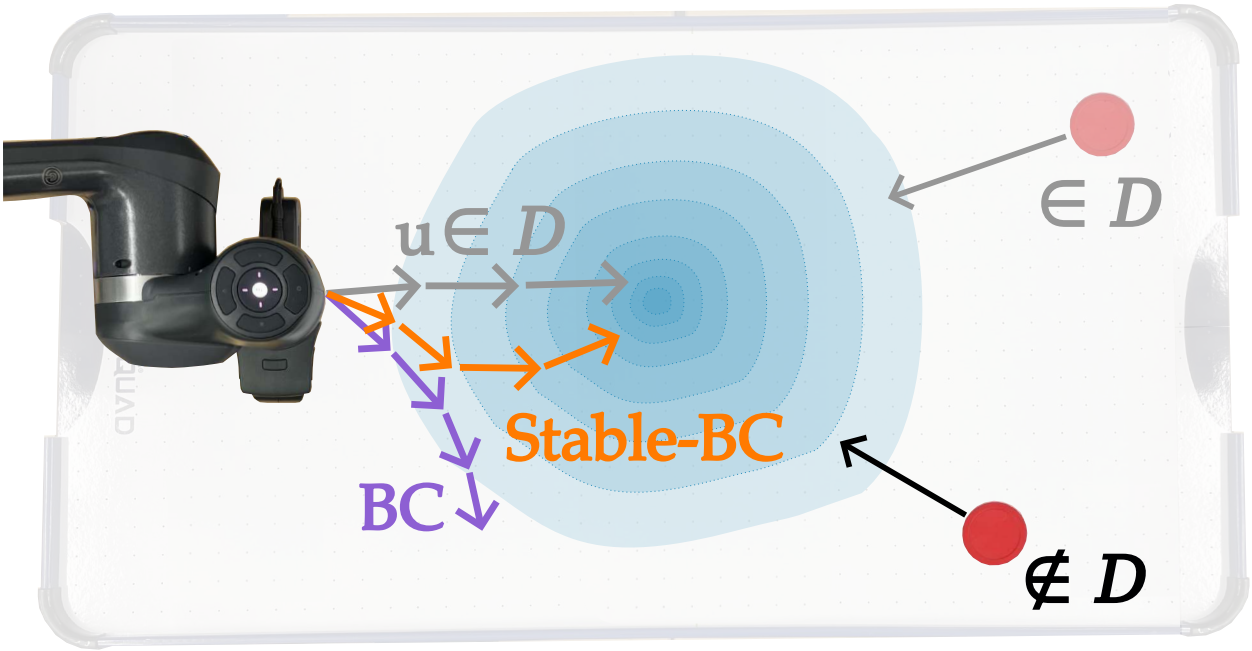


Figure 5.1: Robot playing air hockey by behavior cloning demonstrations $\mathcal{D}$. The robot can successfully hit the puck when it moves at an angle and velocity observed during training ($\in \mathcal{D}$). However, when the puck moves at new angles or velocities ($\notin \mathcal{D}$), standard behavior cloning (BC) misses the puck entirely because of covariate shift. To address this problem we introduce Stable-BC, a variant of BC that encourages the system state to evolve similarly to the expert's demonstrated behaviors.

training distribution, and so we can avoid these errors by intentionally attracting the robot back towards the dataset. We apply this hypothesis to formulate covariate shift as a dynamical system, and derive stability conditions that cause the robot to converge towards behaviors the human has demonstrated. In practice, this results in a behavior cloning algorithm with two loss functions. First, the standard loss that causes the robot policy to match the human's behavior, and second, a stability loss that makes the expert dataset a basin of attraction. Robots that execute this learned policy are inherently robust to covariate shift: e.g., when the puck's angle and velocity in Figure 5.1 deviates from the dataset, the robot extrapolates from the expert examples while still remaining similar to these demonstrated behaviors.

5.2 Problem Statement

We consider settings where a robot is learning to imitate human behaviors. This includes scenarios where the robot is acting in isolation (e.g., a robot arm reaching a goal position), as well as interactive tasks where the robot must reason over other agents (e.g., an autonomous vehicle at an intersection with a human-driven car). Offline the robot is provided with a dataset of state-action pairs, and the robot applies behavior cloning to learn its policy. Our objective is for the robot to extrapolate from the dataset to the learned policy, so that online — when the robot executes this policy — the system is robust to covariate shift.

Robot. Let $x \in \mathcal{X} \subset \mathbb{R}^m$ be the state of the robot and let $u \in \mathcal{U} \subset \mathbb{R}^n$ be the robot's action. For instance, within our air hockey example from Figure 5.1, x is the position of the paddle and u is the robot's end-effector velocity. The robot's state evolves according to the dynamics $\dot{x}(t) = f\big(x(t), u(t)\big)$. We assume that the robot can observe its state x, and that the robot has an accurate model of its own dynamics f.

Environment. During each task the robot interacts with its environment. This environment could consist of static goals, dynamic objects, or even other agents. We separate the state of the environment from the state of the robot: let $o \in \mathcal{O} \subset \mathbb{R}^d$ be the environment's state. Returning to our air hockey example, o could be the angle and velocity of the puck the robot is trying to block. The environment state o updates according to its dynamics $\dot{o}(t) = g\big(x(t), o(t), u(t)\big)$. We assume that the robot can observe the environment state o, but we *do not assume* that the robot has access to the environment dynamics g. For instance, when the robot arm moves to block the puck, the robot does not know how the angle and velocity of the puck might change after collision.

Behavior Cloning. The robot is given a dataset $\mathcal{D}$ of N state-action pairs. Here the overall system state (x, o) consists of both the robot's state and the environment's state. Hence,

the offline dataset is: $\mathcal{D} = \{(x_1, o_1, u_1), \ldots, (x_N, o_N, u_N)\}$.

Based on this dataset, the robot should learn a policy π that maps from system states to robot actions: $\pi(x, o) \rightarrow u$. We instantiate this policy as a neural network with weights θ. Within standard behavior cloning algorithms the robot learns θ such that the policy's actions match the human's actions across states in the dataset. More specifically, the robot learns θ to minimize the loss function:

$$\mathcal{L}_{BC}(\theta) = \sum_{(x,o,u)\in\mathcal{D}} \|\pi_\theta(x, o) - u\|^2 \tag{5.1}$$

Although the resulting policy is designed to mimic the human at states within dataset $\mathcal{D}$, it may fail to match the human when it encounters states outside of this dataset.

5.3 Stable Behavior Cloning

Our objective is to learn a robot policy that maintains the correct behavior despite covariate shift. Here we return to our hypothesis: *behavior cloning is robust when robots stick to what they know*, i.e., when robots remain close to the example motions the expert has demonstrated. We will apply this hypothesis to introduce *Stable-BC*, a control theoretic approach for behavior cloning. Stable-BC is based on the error dynamics between the system's current state and the states in dataset $\mathcal{D}$. We define these error dynamics in Section 5.3.1, and derive the stability conditions under which the error locally converges to zero. In practice, robot policies that satisfy these stability conditions update the robot's state to converge towards behaviors in the dataset, mitigating the effects of covariate shift. We next apply our stability analysis to derive new loss functions for behavior cloning when the robot has access to a model of the environment (Section 5.3.2), and when the environment dynamics are unknown

(Section 5.3.3).

5.3.1 Error Dynamics and Stability Analysis

We start by formulating the error dynamics between the current state of the system and the states observed during training. Let (x', o') be the current state, and let $(x, o) \in \mathcal{D}$ be a labeled state from the training dataset. In order to remain close to behaviors that the human expert has demonstrated, the robot should take actions so that a trajectory starting at (x', o') converges towards a trajectory starting at (x, o). Recall that the robot state x updates according to the dynamics $\dot{x} = f(x, u)$, and the environment state o updates according to dynamics $\dot{o} = g(x, o, u)$. Utilizing these equations, the overall error dynamics become:

$$\dot{x}' - \dot{x} = f\big(x', u'\big) - f\big(x, u\big) \tag{5.2}$$

$$\dot{o}' - \dot{o} = g\big(x', o', u'\big) - g\big(x, o, u\big) \tag{5.3}$$

Compounding errors occur when (x', o') diverges from the dataset $(x, o) \in \mathcal{D}$, i.e., as the error grows over time. To mitigate compounding errors we therefore want to design the robot's policy such that Equations (5.2) and (5.3) converge to zero. Substituting $u = \pi(x, o)$ into the above equations, where π is the robot's policy, and applying a first order Taylor Series approximation around $(x, o) \in \mathcal{D}$, we reach:

$$\begin{aligned}\dot{x}' - \dot{x} = \big(\nabla_x f + \nabla_u f \cdot \nabla_x \pi\big)(x' - x) \\ + \big(\nabla_u f \cdot \nabla_o \pi\big)(o' - o)\end{aligned}$$

$$\dot{o}' - \dot{o} = \big(\nabla_x g + \nabla_u g \cdot \nabla_x \pi\big)(x' - x)$$
$$+\big(\nabla_o g + \nabla_u g \cdot \nabla_o \pi\big)(o' - o)$$

It is important to recognize that these equations are *coupled*. The covariate shift in the robot state x is a function of the covariate shift in the environment state o, and vice versa. This coupling is reflected in our motivating example from Figure 5.1: if the puck moves with a new angle or velocity such that o' is dissimilar from any o in the dataset (i.e., $\|o' - o\|$ is large), then this could cause the robot's position x' to increasingly diverge from labeled states x.

Below we write the coupled system in standardized form:

$$\dot{z} = Az \tag{5.4}$$

where z is the augmented error state, and A is a square matrix that captures the coupled error dynamics:

$$z = \begin{bmatrix} x' - x \\ o' - o \end{bmatrix}, \tag{5.5}$$

$$A = \begin{bmatrix} \nabla_x f + \nabla_u f \cdot \nabla_x \pi & \nabla_u f \cdot \nabla_o \pi \\ \nabla_x g + \nabla_u g \cdot \nabla_x \pi & \nabla_o g + \nabla_u g \cdot \nabla_o \pi \end{bmatrix} \tag{5.6}$$

Ideally, we want the error dynamics $\dot{z} = Az$ to be *stable* about the equilibrium $z = 0$. If we achieve this stability, then the the behavior starting at the current system state (x', o') will converge towards the demonstrated behavior starting at a labeled state (x, o). This mathematically formalizes our original hypothesis: stabilizing Equation (5.4) encourages the robot to take actions that remain close to the examples the expert has demonstrated.

Because matrix A is a local, linearized approximation of the error dynamics, we conclude that $\dot{z} = Az$ is *locally* stable if and only if matrix A is stable, i.e., if all eigenvalues of A have negative real parts [156] (Theorem 7.1). When A is stable the system locally converges towards $z = 0$. The rate of convergence is determined by the eigenvalues of matrix A, where more negative eigenvalues result in faster convergence [156] (Theorem 7.2).

5.3.2 Stable-BC for Model-Based Settings

Overall, our analysis from Section 5.3.1 indicates that we can mitigate errors between the system's current behavior and the expert's demonstrated behaviors by ensuring that matrix A is stable. Inspecting Equation (5.6), we find that A depends upon the robot dynamics f, the robot policy π, and the environment dynamics g. In this section we will focus on *model-based settings* where the robot has access to all of these terms. Put another way, here we assume that the robot not only knows its own dynamics $f(x, u)$, but it also has an accurate model of the environment dynamics $g(x, o, u)$.

Stability vs. Performance. Within model-based settings the robot can directly compute the A matrix. Accordingly — in order to make the matrix A locally stable — we simply need to train the robot's policy π such that all the eigenvalues of A have negative real parts across each state $(x, o) \in \mathcal{D}$. But just ensuring that A is stable does not mean that the robot has learned to perform the task correctly. In fact, this stability can conflict with performance; for instance, when A is stable the robot may converge to $z = 0$, and then remain at rest at that local equilibrium instead of continuing to complete the task. In practice, we resolve this theoretical conflict between stability and performance by training the robot to match the expert's demonstrations (standard behavior cloning loss) while also penalizing the robot's

policy when A is unstable (our proposed addition). This leads to the loss function:

$$\mathcal{L}(\theta) = \sum_{(x,o,u)\in\mathcal{D}} \Bigg[\|u - \pi_\theta(x,o)\|^2 + \lambda \sum_{\sigma_i \in eig(A)} ReLU(Re(\sigma_i)) \Bigg] \tag{5.7}$$

The first term in Equation (5.7) matches the original behavior cloning loss function from Equation (5.1). Within the second term, $\sigma_i \forall i \in \{1, 2, \cdots\}$ are the eigenvalues of A, $Re(\sigma)$ represents the real part of eigenvalue σ, and $ReLU$ is the Rectified Linear Unit activation. The constant $\lambda > 0$ is a hyperparameter set by the designer that determines the relative weight of the two loss terms. In our model-based experiments, we choose $\lambda = 10^{-4}$. Intuitively, a robot policy that minimizes Equation (5.7) mimics the expert's actions across the dataset $\mathcal{D}$, while also shaping its policy to converge towards the demonstrated behaviors. We directly use this loss function to train Stable-BC policies in model-based settings where the robot has an estimate of g.

5.3.3 Stable-BC for Model-Free Settings

In practice, often robots do not have a model of how their actions will impact the environment around them. In this section we therefore consider *model-free settings* where the robot is not given the environment dynamics $g(x, o, u)$. Model-free settings are challenging because, without a model of g, the robot can only compute the top row of the A matrix in Equation (5.6). In general, if we have no information about the environment dynamics g or state o', we cannot make guarantees about the overall stability of matrix A.

Bounded Stability. To resolve this issue we define $\|o' - o\|$ as the magnitude of the environment's covariate shift. In many settings it is reasonable to assume that this magnitude

is *bounded*, i.e., the environment in which the robot is performing its task is similar to the environments seen during training. Moving forward, we will therefore assume that the magnitude of the environment's covariate shift has some upper bound $\|o' - o\| \leq \epsilon$. Let $A_1 = \nabla_x f + \nabla_u f \cdot \nabla_u \pi$ and $A_2 = \nabla_u f \cdot \nabla_o \pi$. Substituting these terms in Equation (5.6), the first row of the A matrix can be written as:

$$\dot{\mathrm{e}}_x = A_1 \mathrm{e}_x + A_2 \mathrm{e}_o, \quad \mathrm{e}_x = x' - x, \quad \mathrm{e}_o = o' - o \tag{5.8}$$

Integrating both sides of this ordinary differential equation, we obtain: $\mathrm{e}_x(t) = \mathrm{e}_x(0)e^{A_1 t} + \int_{\tau=0}^{t} A_2 \mathrm{e}_o(\tau) e^{A_1(t-\tau)} d\tau$. Taking the matrix norm of the result, and substituting in the upper bound for $\|\mathrm{e}_o\| \leq \epsilon$, we obtain an upper bound on the covariate shift in the robot's state:

$$\|\mathrm{e}_x(t)\| \leq \|\mathrm{e}_x(0)e^{A_1 t}\| + \|A_2\|\epsilon \int_{\tau=0}^{t} \|e^{A_1(t-\tau)}\| d\tau \tag{5.9}$$

Intuitively, Equation (5.9) provides an limit on how far the robot state at test time, x', could diverge from the robot states during training, x. Similar to our approach from Section 5.3.1, our goal here is to minimize the upper bound on this error and cause $\|\mathrm{e}_x\|$ to converge towards zero.

In order to minimize the right side of Equation (5.9) we propose to design the robot's policy π such that matrix A_1 is stable. This causes $\|\mathrm{e}_x(0)e^{A_1 t}\| \to 0$, and thus Equation (5.9) simplifies to $\|\mathrm{e}_x(t)\| \leq \|A_2\|\epsilon \int_{\tau=0}^{t} \|e^{A_1(t-\tau)}\| d\tau$. Next, leveraging the properties of matrix exponentials, we have that $\|e^{A_1 t}\| \leq e^{\|A_1\| t}$, where $\|A_1\|$ is the induced 2-norm of matrix A_1. Substituting this inequality back into the equation and solving the integral, we finally reach:

$$\|\mathrm{e}_x(t)\| \leq \frac{\|A_2\| \cdot \epsilon}{\|A_1\|} \cdot \left(e^{\|A_1\| t} - 1\right) \tag{5.10}$$

Algorithm 3 Stable-BC

Given: state-action pairs $\mathcal{D}$ and robot dynamics f
Initialize: robot policy $\pi_\theta(x, o)$ with weights θ
for $i \in 1, 2, \cdots$ **do**
    if g is available **then**
        Compute loss $\mathcal{L}(\theta)$ using Equation (5.7)
    else
        Compute loss $\mathcal{L}(\theta)$ using Equation (5.11)
    end if
    Update robot policy $\theta \leftarrow \theta - \alpha \nabla_\theta \mathcal{L}(\theta)$
end for
return Trained robot policy $\pi_\theta(x, o)$

Equation (5.10) offers a useful upper bound on the robot's state error. Provided that the change in the environment state is bounded by ϵ, Equation (5.10) shows that the covariate shift in the robot's state is also bounded, and the magnitude of that bound depends on the off-diagonal matrix A_2. For instance, if we design the matrix A such that $\|A_2\| \to 0$, then the upper bound on $\|\mathrm{e}_x(t)\|$ also converges towards zero, and the robot's behavior at test time will remain similar to the examples given at training time.

Stability vs. Performance. To summarize our analysis, in model-free settings we cannot directly stabilize matrix A. Instead, we enforce an upper bound on the covariate shift in the robot's state by designing policy π such that:

1. All eigenvalues of matrix A_1 (the top left component of A) have negative real parts
2. The magnitude of matrix A_2 (the top right component of A) is minimized

We note that the robot can compute both A_1 and A_2, since neither term depends on the environment dynamics g.

Examining these two conditions we again find a conflict between stability and performance.

Specifically, if $\|A_2\| \to 0$, then $\nabla_o \pi \to 0$ and the robot's policy no longer depends upon the environment state o. From a stability perspective, this is desirable because $\|A_2\| = 0$ means that the equations in Equation (5.4) are decoupled, and thus any error $o' - o$ will not impact $x' - x$. From a performance perspective, however, this is undesirable because we often need the robot to make decisions based on its environment state. Consider our air hockey example: to successfully hit the puck, the robot's policy must reason over the state of that puck. Similar to Section 5.3.2, we practically resolve this conflict by training a robot policy that trades-off between mimicking the expert's actions and satisfying the two stability conditions. Our loss function for Stable-BC in model-free settings is:

$$\mathcal{L}(\theta) = \sum_{(x,o,u)\in\mathcal{D}} \Bigg[\|u - \pi_\theta(x,o)\|^2 + \lambda_1 \|A_2\| + \lambda_2 \sum_{\sigma_i \in eig(A_1)} ReLU(Re(\sigma_i)) \Bigg] \tag{5.11}$$

where σ are the eigenvalues of the sub-matrix A_1. The first term in Equation (5.11) matches the original behavior cloning loss function from Equation (5.1). The other terms enforce our two conditions for bounded stability, and λ_1 and λ_2 are hyperparameters selected by the designer. For our model-free experiments, we choose the relative weights of $\lambda_1 : \lambda_2 = 1 : 100$. We note that this result for the model-free case is weaker than in the model-based case. Within Section 5.3.2 we provided conditions for local asymptotic stability in both e_x and e_o; by contrast, here we can only ensure bounded stability for e_x assuming an upper bound on the magnitude of e_o.

Algorithm Summary. Given a dataset of state-action pairs $\mathcal{D} = \{(x_1, o_1, u_1) \ldots (x_N, o_N, u_N)\}$ and robot dynamics f, the robot uses the procedure outlined in Algorithm 3 to learn a behavior cloned policy that is locally stable around the expert demonstrations. We refer to this approach as Stable-BC. If the robot has access to the environment dynamics g, then

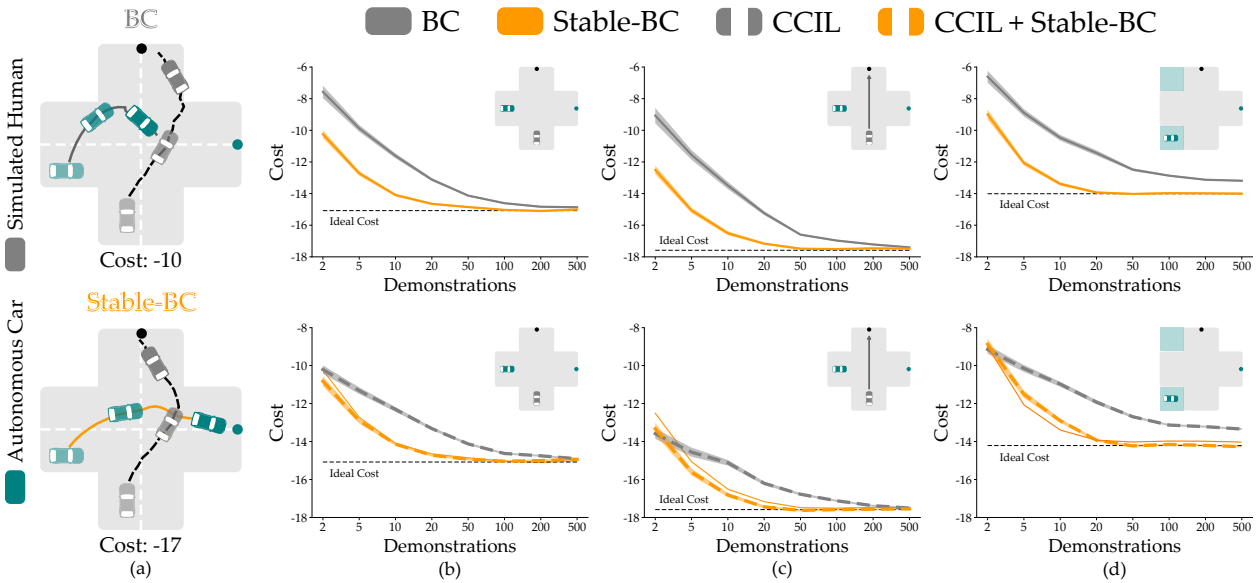


Figure 5.2: Simulation results from interactive driving. (a) An example rollout using BC and Stable-BC. With BC the autonomous car gets stuck in the middle of the intersection. By contrast, when using Stable-BC the autonomous car lets the human pass and then crosses afterwards, resulting in a lower cost. (b-d) Average cost over 100 trials as a function of the number of expert demonstrations. In (b) the testing environment matches the training environment. In (c) the human agent ignores the autonomous car, and in (d) the autonomous car starts from initial states outside of its training distribution. Shaded regions show SEM. Ideal cost is the best-case scenario where the autonomous car's learned policy exactly matches the policy of the human teacher. In the bottom row we plot Stable-BC (solid orange) and CCIL + Stable-BC (dashed orange).

we use Equation (5.7) as the loss function. Alternatively, if the environment dynamics are unknown, the robot leverages Equation (5.11) as the loss function.

5.4 Simulations

In Section 5.3 we presented Stable-BC, our method for shaping behavior cloned policies such that they have stable error dynamics. Next we will perform controlled simulations that compare Stable-BC to standard behavior cloning and recent off-policy variants. We

consider three different tasks: (Section 5.4.1) an interactive driving environment where an autonomous car and human vehicle are trying to cross an intersection, (Section 5.4.2) a single-agent quadrotor environment where drone with nonlinear dynamics must safely navigate 3D spaces, and (Section 5.4.3) a simple visual setting where a point mass uses RGB images to estimate its goal position. The code for implementation can be found here: https://github.com/VT-Collab/Stable-BC

5.4.1 Interactive Driving

In our first simulation an autonomous car learns to cross an intersection while avoiding a human driver (see Figure 5.2). This environment is challenging for the autonomous car because it is *interactive*: even if the autonomous car matches the demonstrated behavior, changes in how the human drives during policy execution can lead to covariate shift.

Environment. The autonomous car's state $x \in \mathbb{R}^2$ is its position, and action $u \in \mathbb{R}^2$ is the autonomous car's velocity. State x updates with the known linear dynamics $\dot{x} = u$. During each interaction the autonomous car tries to reach a static goal position on the opposite side of the intersection while maintaining a safe distance from a human driver. Here $o \in \mathbb{R}^2$ is the position of the human's vehicle, and o evolves with unknown and nonlinear dynamics $g(x, o)$. The autonomous car is given a dataset $\mathcal{D}$ of offline demonstrations. In each demonstration two simulated humans show how both vehicles should navigate the intersection. The initial car positions x and o are uniformly randomly sampled from regions on the left and bottom of the intersection. Then one simulated human expert drives the autonomous car while noisily

optimizing the following cost function:

$$Cost(x, o, c) = \|x(t + \Delta t) - c\| - \|x(t) - c\| + 0.75 \cdot \|x(t) - o(t)\| - 0.75 \cdot \|x(t + \Delta t) - o(t)\| \quad (5.12)$$

where c is the constant goal position. The first two terms of Equation (5.12) encourage the car to move towards goal c, and the final two terms penalize actions that get closer to the other vehicle o. Simultaneously, a second simulated human controls the human-driven car while optimizing the same cost function (where x and o are switched). Each individual demonstration results in 20 state-action pairs (x, o, u).

Methods. The autonomous car learns from these demonstrations using four different methods. We start with standard behavior cloning (**BC**) trained using Equation (5.1). Next, we implement **CCIL** [22]: CCIL is a state-of-the-art off-policy approach that builds a dynamics model from the dataset, and then leverages that model to synthetically increase the number of expert state-action pairs. We compare these baselines to our approach applied independently (**Stable-BC**), as well as our approach applied alongside CCIL. In **CCIL + Stable-BC** we first use CCIL to enhance the offline dataset, and then train Stable-BC on this augmented dataset.

We also compare **Stable-BC** to an Offline-RL algorithm **CQL** [78]. The detailed implementation of **CQL** and results for this comparison are provided in our GitHub repository.

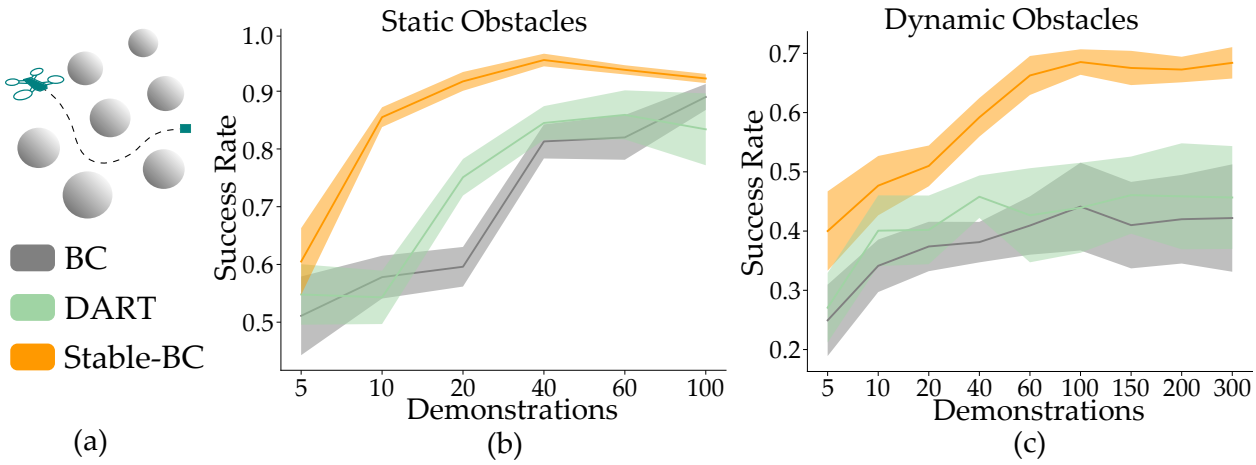
Results. Our results are summarized in Figure 5.2. We report the autonomous car's total cost across an interaction, where the cost at the current timestep is computed using Equation (5.12). In the top row we compare BC to Stable-BC, and in the bottom row we compare CCIL and CCIL + Stable-BC. We also plot Stable-BC in the bottom row for reference.

To evaluate the robustness of the learned policies, we executed each trained policy in three different testing environments. We started with a testing environment that exactly matched the training environment (left column). Next, we modified the dynamics g of the human-driven car (middle column). Instead of moving to avoid the autonomous car with dynamics $g(x, o)$, now the simulated human was self-centered, and only reasoned over their own state using dynamics $g(o)$. Finally, we sampled the autonomous car's initial state $x(0)$ from regions outside of the training distribution (right column). The human again used the training dynamics $g(x, o)$, but the autonomous car had to navigate around that human from new regions of the workspace.

Summary. Across all testing environments, our results indicate that Stable-BC outperforms BC and CCIL. This result suggests that the change in performance is not because Stable-BC is overfitting to the training data, as evidenced by its performance in out-of-distribution environments. We also observe that the differences between Stable-BC and CCIL + Stable-BC are negligible. This is a positive result because it suggests that our approach is more robust to covariate shift in interactive settings, and that off-policy data-augmentation methods may not be necessary when applying Stable-BC.

5.4.2 Nonlinear Quadrotor Navigation

In this simulation, we assess the performance of Stable-BC within a more complex, nonlinear, high-dimensional (3-D) system. Specifically, we consider a quadrotor that must navigate across a room populated with spherical obstacles (see Figure 5.3). The quadrotor's state $x \in \mathbb{R}^6$ includes its position (p_x, p_y, p_z) and velocity (v_x, v_y, v_z), and the quadrotor's action $u \in \mathbb{R}^3$ includes its acceleration u_T, roll u_ϕ, and pitch u_θ. State x evolves with nonlinear



(a) (b) (c)

Figure 5.3: Simulation results for nonlinear quadrotor navigation. (a) An example trajectory of the quadrotor flying around the 3D obstacles to reach its goal position. (b) Average success rate with constant obstacles positions. (c) Average success rate when obstacle positions change across interactions. We trained the system end-to-end 10 separate times, and then performed 100 test rollouts with each trained model. Shaded regions show SEM.

dynamics:

$$\dot{p}_x = v_x, \quad \dot{p}_y = v_y, \quad \dot{p}_z = v_z$$

$$\dot{v}_x = a_g \tan u_\theta, \quad \dot{v}_y = -a_g \tan u_\phi, \quad \dot{v}_z = u_T - a_g$$

We examine two scenarios: (1) static obstacles and (2) randomly positioned obstacles in each episode. For the first case, the quadrotor's control depends only on its state x. Here, the obstacle positions are implicitly contained in x, as both the environment layout and target location remain constant. In the case of dynamically placed obstacles, we explicitly introduce an environment state $o \in \mathbb{R}^{14}$, representing the obstacles' positions in the yz-plane. At the start of each episode, the quadrotor is initialized randomly on one side of the room, aiming to reach a fixed target on the opposite side while avoiding seven obstacles. The episode concludes successfully if the quadrotor reaches within 0.5 units of the target or terminates in failure upon colliding with an obstacle or room boundary.

Methods. For both cases, we compare Stable-BC to two baselines: **BC** and **DART** [79]. DART is a state-of-the-art data collection approach that perturbs the expert while they provide demonstrations to increase dataset diversity. As the robot collects expert demonstrations offline, DART iteratively estimates the errors between the expert's actions and its current policy. DART then injects noise based on these errors when collecting new demonstrations from the expert; this causes the expert to show the robot more diverse and corrective behaviors. BC and Stable-BC are trained using the same offline dataset that does not include DART's perturbation procedure. To test the robustness of the learned policies and simulate real-world conditions, we inject Gaussian noise into quadrotor's actions at test time.

Results. Our results are summarized in Figure 5.3, where we report the success rate, defined as the fraction of trials in which the quadrotor reached its goal without colliding. For all methods, the success rate increases when the robot is given more expert demonstrations. In both scenarios, we observe that Stable-BC achieves a higher success rate with fewer demonstrations as compared to the baselines. Specifically, while DART relies on perturbed expert demonstrations to collect more diverse data, Stable-BC converges to best-case performance more rapidly using only the original offline dataset. These results demonstrate that Stable-BC can be effectively applied to nonlinear systems.

5.4.3 Point Mass with Visual Observations

Our final simulation examines whether Stable-BC extends to visual settings. Here a point mass robot attempts to reach a 2D goal location. The robot's state $x \in \mathbb{R}^2$ is its position, and x updates with linear dynamics $\dot{x} = u$. In each interaction the start and goal position are uniformly randomly sampled. However, the robot is not given direct access to the goal;

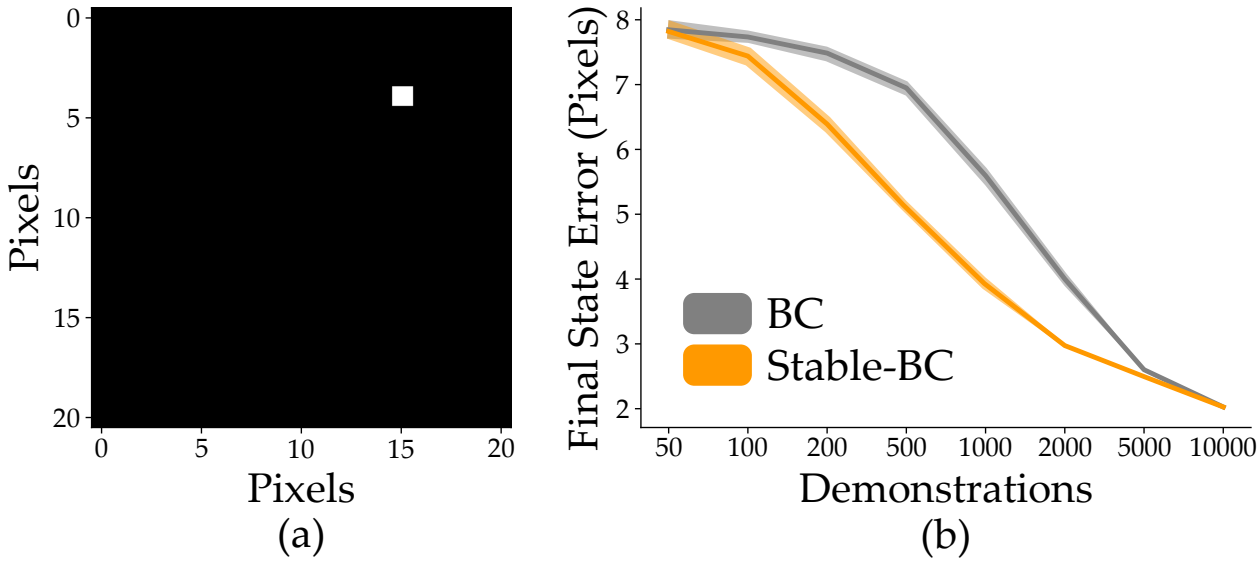


Figure 5.4: Simulation results for visual observations. (a) The robot is trying to reach a goal. At each timestep the robot observes image o where the goal position is marked by a white pixel; here we show an example of one of these images. The goal position and robot position are randomly sampled at the start of each new interaction. (b) Average distance between the goal and the robot's final position over 25 trials. Shaded regions show SEM.

instead, the robot observes an image o that displays the goal location in pixel space (see Figure 5.4). The robot must learn a policy that moves towards this goal based on its current position x and the visual observation o.

Methods. The images that the robot receives have 21×21 pixels. Offline, a simulated expert shows the robot what actions to take in respond to these images: each demonstration consists of a $(x,\ o,\ u)$ pair (i.e., at state x, if the robot observes image o, it should take action u). After collecting the set of these demonstrations, we first train an autoencoder that embeds images o into a 10-dimensional latent space using the encoder $\mathcal{E}(o)$. We then apply **BC** and **Stable-BC** to learn policies of the form $\pi(x, \mathcal{E}(o))$. Both methods are trained using the same demonstration data.

Results. In Figure 5.4 we plot the distance between the robot's position at the end of

each interaction and the goal state (i.e., Final State Error). If the robot moves completely randomly, the expected Final State Error is 10 units. Our results from this proof-of-concept simulation suggest that Stable-BC can be applied to settings where o consists of visual observations; we find that Stable-BC outperforms standard BC when given the same amount of training data.

5.5 Air Hockey Experiment

In this section we evaluate Stable-BC in a real-world environment with user-provided training data. Specifically, we conduct imitation learning experiments where participants teach a 7-DoF Franka Emika robot arm to play a simplified game of air hockey. We compare Algorithm 3 (Stable-BC) to standard behavior cloning (BC). Videos of our air hockey experiments are available here: https://youtu.be/ZC3BjY1k18w

Experimental Setup. The robot's task is to hit the hockey puck so that it bounces off the opposite side of the table and returns to the robot (see Figures 5.1 and 5.5). The robot's state $x \in \mathbb{R}^2$ is the position of its end-effector on the surface of the air hockey table, and action $u \in \mathbb{R}^2$ is the robot's end-effector velocity. A camera is mounted directly above the table to track the position of the hockey puck at a frame rate of 20 Hz. The environment state $o \in \mathbb{R}^4$ is the current and previous position of the puck in this camera frame; o evolves with unknown dynamics $g(x, o, u)$. Because the robot does not have access to g, in this experiment we applied the model-free version of our proposed Stable-BC algorithm.

Training Data. We recruited 10 members of the Virginia Tech community to provide offline training data. Participants gave their informed consent under IRB #23-784. We first gave the participants 2 minutes to practice controlling the robot and hitting the puck. Once this practice was complete, each participant teleoperated the robot to repeatedly hit the puck

against the opposite side of the table for ~ 2.5 minutes. This resulted in ~ 3000 state-action pairs per user. We kept each user's data separate, so that we obtained 10 different datasets $\mathcal{D}_1 \dots \mathcal{D}_{10}$ that we used to test our approach.

Testing Procedure. Given the expert datasets $\mathcal{D}_1 \dots \mathcal{D}_{10}$, we trained robot policies using BC and Stable-BC. We varied the amount of data the robot had access to during training — e.g., we trained robot policies with 15, 60, and 120 seconds of expert data. For each amount of training data we learned 10 different policies (one for every user's dataset), and then we tested the performance of each policy across 10 independent rollouts. The proctor started every trial by pushing the puck towards the robot, and then the robot executed its policy to autonomously and repeatedly hit the puck.

We quantified the performance of the robot learner by measuring the average number of times that the robot consecutively hit the puck against the opposite side of the table without missing it (*Number of Successful Hits*). If the robot successfully hit the puck 25 times in a row, we terminated the trial there; i.e., 25 was the maximum possible number of successful hits. However, successfully hitting the puck does not capture the quality of the robot's motion. To evaluate the quality of the robot's motion, we also measured the *Number of Direction Changes* per successful hit. A direction change was defined as a difference between actions u^t and u^{t-1} of more than 10 degrees.

Hypothesis. We had the following two hypotheses:

H1. *Given the same training data, Stable-BC will achieve a higher number of successful hits as compared to BC.*

H2. *Stable-BC will learn policies that output actions with fewer direction changes.*

Results. Our results are summarized in Figure 5.5. A repeated measures ANOVA revealed that both the robot's learning algorithm ($F(1, 9) = 19.03$, $p < 0.05$) and the amount of

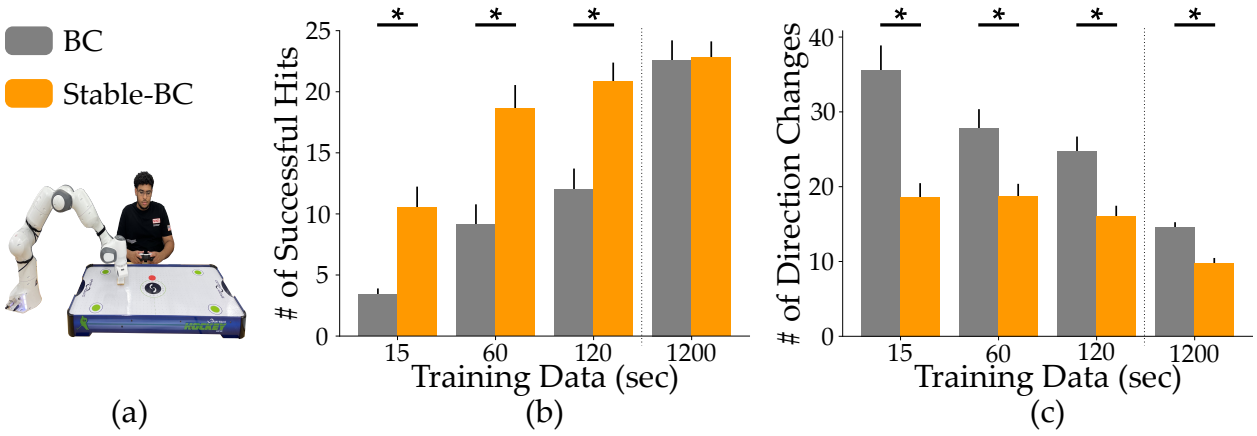


Figure 5.5: Results for the air hockey experiment in Section 5.5. (a) Participants teleoperated a 7 DoF robot arm to hit the puck. We collected their demonstration data offline, and then used this data to train BC and Stable-BC policies. (b) We measured the number of successful hits with different amounts of training data. Ideally, a robust robot policy will repeatedly hit the puck, even when that puck travels with previously unseen angles and velocities. Both BC and Stable-BC eventually converged to equivalent performance, but Stable-BC reached that performance with a smaller amount of training data. (c) To qualitatively assess the learned behavior, we also measured the number of direction changes per successful hit. Stable-BC produced policies that were more smooth and consistent, with fewer direction changes than BC. Error bars show SEM and $*$ denotes statistical significance ($p < 0.05$).

training data ($F(2, 18) = 50.79$, $p < 0.05$) had significant effects on the number of successful hits. For 15, 60, and 120 seconds of training data, Stable-BC resulted in more robust policies that had a higher number of successful hits than BC ($p < 0.05$). As expected, the performance of both imitation learning algorithms increased in proportion to the amount of training data. But Stable-BC was able to converge to ideal performance with less data than BC: under Stable-BC, the number of successful hits with 120 seconds (2 minutes) of data was only marginally less than the number of successful hits with 1200 seconds (20 minutes) of training data. Both BC and Stable-BC converged to similar performance when given 1200 seconds of data — i.e., the combined data across all 10 users — indicating that Stable-BC is as effective or more effective than BC across all data levels. Overall, these results support hypothesis **H1**.

We next explored the smoothness of the robot's learned policy. As before, a repeated measures ANOVA found that the robot's learning algorithm ($F(1, 9) = 59.65$, $p < 0.05$) as well as the amount of training data ($F(2, 18) = 9.62$, $p < 0.05$) had significant effects of the number of direction changes. Post-hoc analysis confirmed that across all levels of learning data, Stable-BC produced policies with significantly fewer direction changes ($p < 0.05$) than BC. We even observed that Stable-BC had fewer direction change when trained on the combined dataset with 1200 seconds of data ($t(9) = 4.681, p < 0.05$). Viewed together, these results support hypothesis **H2** and suggest that not only does Stable-BC lead to more robust policies, but these policies are qualitatively more consistent.

5.6 Conclusion

In this paper we presented a behavior cloning approach to reduce covariate shift. Instead of focusing on the training data, our method explored on the error dynamics between the

robot's current behavior and the expert's demonstrated behaviors. By performing control theoretic analysis on these dynamics, we derived model-based and model-free stability conditions for shaping the learned policy to bound covariate shift. Our resulting algorithm, Stable-BC, is an easy to implement extension of standard behavior cloning that can be used independently or alongside existing data-centric approaches. Multiple experiments across interactive, nonlinear, visual, and real-world environments suggest that Stable-BC produces more robust policies than state-of-the-art baselines given the same training data.

Future Works. Moving forward, we plan to explore how this work can be extended to Offline-RL settings. We recognize that Offline-RL has access to reward functions when learning from human demonstrations. Further work is needed to understand how the reward information can be leveraged in the stability analysis and how it would affect the stability conditions of our approach.

Chapter 6

Robot Learning from 2D Drawings

In the previous chapters, we have discussed how can we make it easier for the robots to learn robust policies from human demonstrations. However, in all of these approaches, the user needs to provide demonstrations, either by physically guiding the robot through the task or by performing the task themselves. While doing so, the user needs to provide multiple demonstrations ensuring that numerous task scenarios are covered in order for the robot to generalize to the diverse task settings. This makes the process of collecting demonstrations tedious and expensive. In this chapter, we look at the problem of imitation learning from the users' perspective and develop an algorithm that enables the users to teach the robot by providing drawings on the images of the environment. This approach of teaching using drawings makes the process of providing demonstrations faster and easier, while minimizing the users' physical interactions with the environment.

6.1 Introduction

Robots can learn new tasks by imitating human examples. In general, as humans provide more examples of a task, and the more diverse those examples are, the robot will learn to perform that task more effectively. Consider the task depicted in Figure 6.1, where the robot must scoop cereal from a bowl. One common way for humans to teach this task is by physically demonstrating it, i.e., kinesthetically backdriving the robot's joints through

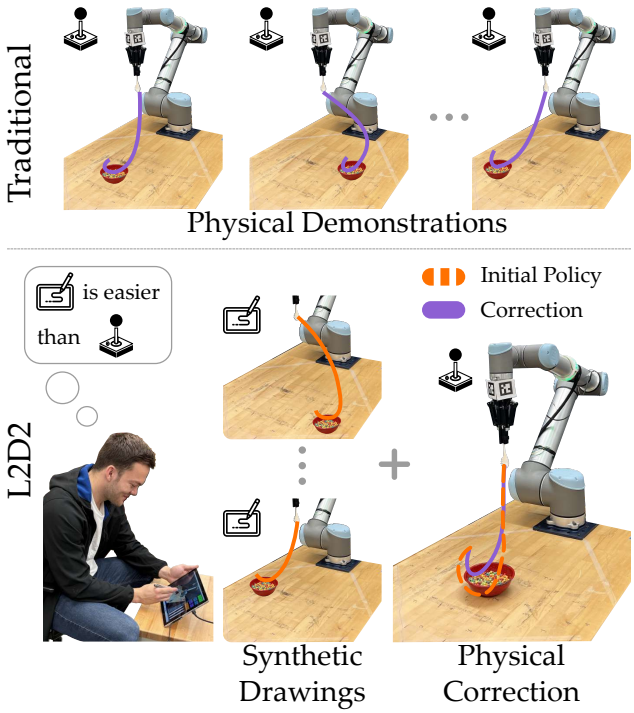


Figure 6.1: Human demonstrating a scooping task to the robot using different teaching paradigms. When using traditional methods to teach the robot, the user needs to manually reset the environment by changing the bowl position and provide demonstrations by physically guiding the robot through the task. We propose L2D2, an approach that synthetically generates diverse environment settings and enables the human to demonstrate the task by drawing a trajectory on the artificial images of the environment. If the robot makes a mistake when executing the learned task, the user provides a few physical corrections to fine-tune the robot's learned behavior. Our proposed approach reduces costly physical interactions and enables humans to teach robots efficiently.

the process of reaching the bowl and rotating the spoon. The robot can simply copy this motion if the bowl always stays in the same place. But for the robot to learn how to scoop cereal when the bowl is moved, the human needs to demonstrate the task for various bowl positions. This process of teaching the robot can not only be challenging — the human needs to carefully orchestrate the motion of the robot — but is also time-consuming. Each time the human wants to show another example, they will have to reset the robot, place the bowl in a new spot, and demonstrate the task again. So how can we make it easy for humans to provide these diverse examples?

Recent works have explored methods that simplify the teaching process by enabling humans to conveniently provide informative demonstrations. For example, humans can control the robot via remote teleoperation, making the process as intuitive as performing the task with their own hands [9, 102, 103]. Alternatively, humans can take a video where they perform the task themselves and provide language descriptions that explain their actions [27, 28, 110, 111]. These works follow a general trend: 1) making it easier for humans to perform the task and 2) accurately capturing the task complexities through real-world demonstrations. However, for these existing approaches to work, the human still needs to perform the task either vicariously or with their own body while physically changing the environment for every example they demonstrate. Because each iteration of providing an example is laborious, this fundamentally limits how much these approaches can accelerate data collection. Additionally, since these approaches require physical interactions with the system, the human teachers need to have access to the system in order to demonstrate the tasks.

We want to make it easier for humans to teach the robot, while reducing dependence on the access to the physical system. Building on the recent trends, we envision a system where the human can teach the robot by *drawing* the desired task on an image of the environment. Humans can generate drawings rapidly, and — because they are not actually performing the task — the human teacher is not constrained by the physical speed of the learner or the burden of resetting the environment between demonstrations. Since drawings allow the users to demonstrate tasks without interacting with the robot or its environment, it would enable large-scale asynchronous data collection without the need for a physical system. However, we also recognize that drawings as a form of demonstration are themselves limited: when using drawings the human is potentially trying to convey a complex, high-dimensional task on a 2D image of the environment. To bridge the gap between the low-cost, easy to provide drawings and information-rich demonstrations, we hypothesize that:

Robots can rapidly learn new tasks by

combining diverse canonical drawings with a few high-dimensional demonstrations.

We propose **L2D2**: **L**earning from **2D D**rawings, a three-step approach for learning from human teachers. First, the robot takes an image of the environment, and the human annotates task-relevant objects or features that can vary between task iterations. Next, the human teacher iteratively draws the task on the images provided by the robot: at each iteration, we leverage vision and language models to artificially manipulate the positions of the objects in the image and adjust the features that the human highlighted. This approach quickly results in a large dataset of diverse demonstrations. For instance, in our user study, in the time it takes to provide 10 physical demonstrations, users were able to provide $\sim$20 drawings.

In the final step, the robot extracts high-dimensional demonstrations from the drawings, trains a control policy, and then attempts to perform the task in the real world by rolling out the learned policy. If the human teacher sees that the robot is making a mistake when performing the task, they can physically correct the robot to refine its behavior. These corrections help the robot ground the drawings in reality and fill any gaps in information. Returning to our motivating example in Figure 6.1, after observing the diverse drawings, the robot may learn to reach the cup but may not learn to precisely rotate the spoon. The human teacher can demonstrate this rotation at run-time to correct the robot's motion, thus reinforcing the learned policy.

6.2 Problem Statement

We consider settings where a robot learns manipulation tasks from a human teacher. We assume that the robot is equipped with a static camera that can take images of the entire

task environment. Using our proposed approach, users can convey their desired task to the robot by drawing on the image of the environment (using a tablet or a similar device). Below, we identify key differences between learning from drawings and learning from real-world demonstrations. First, the human's drawings are 2-dimensional, but the robot needs to learn manipulation tasks in the 3-dimensional world. Second, in drawings, the robot cannot actually interact with objects in the environment (e.g., the robot cannot pick up a block in a drawing). Viewed together, these differences result in an information gap that our approach must resolve to successfully learn to perform tasks in the real world using the human's 2D drawings.

Setup. Before collecting the human's demonstrations, an RGB camera is placed in the environment. The camera is fixed throughout the training and evaluation process. The camera should be carefully positioned such that it can view the robot arm and its work environment, and the information loss in representing the robot states with 2-dimensional images is minimized (see Section 6.3.2 for further details). Accordingly, we limit our experiments to learning manipulation tasks where the robot's movements lie within the camera's field of view. These manipulation tasks still cover a wide array of object-centric motions like reaching, grasping, pushing, pulling, pick-and-place, pouring, scooping, etc., as well as combinations of these primitive motions.

Robot. A robot arm interacts with the environment to perform the task. At each step of the task, the robot reads the state of its arm $s_R \in \mathbb{R}^d$ and receives an image of the environment from our fixed camera. Consistent with the related works, we leverage language feedback from the users and vision models to extract task-relevant features from the images of the environment [157, 158]. The extracted features $o \in \mathbb{R}^k$ form the state of the environment. The overall state of the system comprises of the state of the robot and the observed features in the environment, represented as $s = (s_R, o)$. Returning to our motivating example, s_R

may be the position and orientation of the robot's end-effector as well as the configuration of its gripper, and o could specify the location of the bowl the robot is trying to reach. When the robot performs a task in the environment, it takes an action $a \in \mathbb{R}^d$, and the state transitions according to system dynamics $\mathcal{T}(s, a)$. We do not assume that the robot has access to these transition dynamics.

Drawings. The human is provided with an image of the environment taken from the fixed camera. Let $p \in \mathbb{R}^2$ be a 2D point on this image. The human draws on this image to demonstrate their desired behavior to the robot. This drawing is provided by the human in the form of a *trajectory* in the 2D space, and comprises a sequence of points $\xi_P = [p_1, p_2, \cdots, p_n]$, overlaid onto the initial image of the environment (see Figure 6.3).

When the human provides a drawing, they are not only thinking about the 2D trajectory (i.e., the path they draw on the image), but also about the analogous d-dimensional trajectory that the robot should follow in the real world (e.g., reaching the bowl and rotating the spoon). We denote this corresponding trajectory in the robot's state-space as $\xi = [s_1, s_2, \cdots, s_n]$. Moving from drawings ξ_P to the real-world trajectories ξ suffers from two fundamental challenges: 1) Since there are an infinite number of mappings from a low-dimensional to a high-dimensional space, the robot does not know *a priori* how to map the trajectories drawn on the image to their real-world counterparts, and 2) the drawings do not capture how the state of the environment changes throughout the task. In other words, while the human's drawings provide information about the trajectory the robot should follow, i.e., $\xi_R = [s_{R_1}, s_{R_2}, \cdots, s_{R_n}]$, they do not convey how the environment state o should evolve.

Objective. Given these challenges, our objective is to develop an interface that enables users to efficiently provide a diverse set of drawings that convey their desired task, and an associated algorithmic framework that can leverage these drawings to learn robot policies.

As a first step towards this goal, we need a procedure for collecting drawings in a variety of task settings while minimizing physical interactions with the environment. We must then convert the drawings provided by the user to real-world robot demonstrations. These demonstrations should be in the form of state-action pairs (s, a), capturing how the system state s changes when the robot takes an action a. In other words, we want to convert the drawings ξ_P provided by the human on images of the environment to a corresponding dataset of demonstrations in the real world $\mathcal{D} = \{(s_1, a_1), (s_2, a_2), \cdots, (s_n, a_n)\}$.

The robot can then leverage this dataset to learn a policy $\pi_\theta(a|s)$ that imitates the behavior that the human demonstrated in their drawings. Our proposed approach is not tied to any specific method for learning from demonstrations; but in our experiments, we leverage Behavior Cloning [76] within our learning framework. Behavior Cloning matches the actions predicted by the policy in states s to the corresponding actions a in the dataset $\mathcal{D}$ by minimizing the following loss function to learn the policy parameters θ:

$$\mathcal{L}_{BC}(\theta) = \sum_{(s,a)\in\mathcal{D}} \|\pi_\theta(s) - a\|^2 \tag{6.1}$$

Given a dataset of diverse and accurate demonstrations, we expect the robot to generalize the learned policy to new environment configurations. In the next section, we introduce our interface for obtaining diverse drawings and present our method for closing the information gap between these drawings and real-world demonstrations to ensure accurate learning.

6.3 L2D2

Our aim is to make it easy for humans to teach robots by enabling them to convey the desired task through sketches. In particular, we want to minimize the need for physical interactions

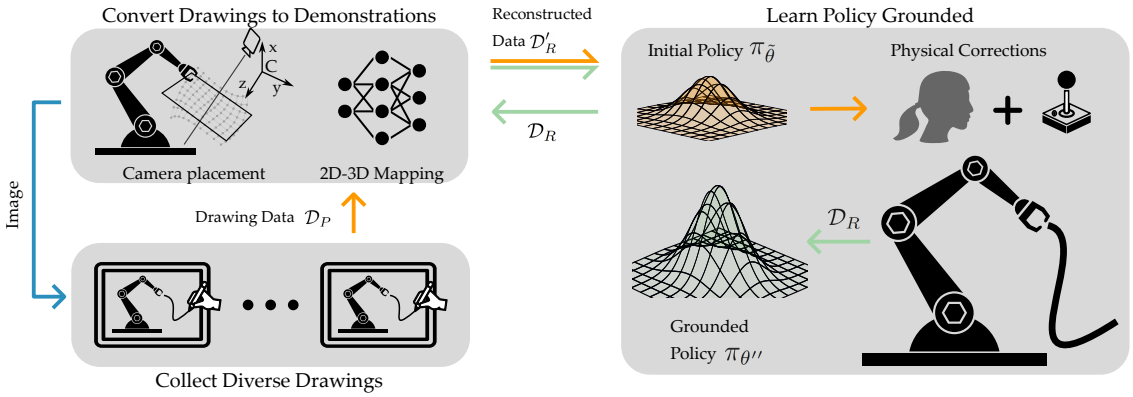


Figure 6.2: Block diagram highlighting the steps in the L2D2 framework. We first start by selecting an optimal camera position in the environment. Using this camera placement, we learn a mapping that maps pixels in the 2D images to the high-dimensional robot state. The arrows in blue color indicate the flow of information in this first step of the process. In the second step (highlighted with orange arrows), an image from this camera position is then passed on to the drawing interface to manipulate the images and collect diverse user drawings. These drawings are then converted to the trajectories in robot's workspace and used to learn an initial policy. The user may then provide physical corrections to the robot which are used to refine the reconstructed demonstrations and ground the policy learned using drawings with the real world information. Green arrows show the steps and flow of information in this stage of L2D2.

with the real world and efficiently collect diverse training data by synthetically generating new task configurations. As detailed in the previous section, we recognize the challenges that come with operating on 2D images instead of the 3D world. In this section, we present our approach for addressing these issues and learning a robust robot policy. First, in Section 6.3.1, we describe our interface for obtaining a diverse set of drawings. Next, we explain how these sketches can be accurately mapped to high-dimensional demonstrations in the real world (see Section 6.3.2). Lastly, in Section 6.3.3, we propose how policies learned from the diverse drawings can be grounded with a few real-world demonstrations to bridge the fundamental gap between static images and dynamic physical interactions. Figure 6.2 shows the progressive relationship between these sections and provides a logical chain of flow from optimizing camera placement to collecting diverse drawings to finally learning a policy using drawings.

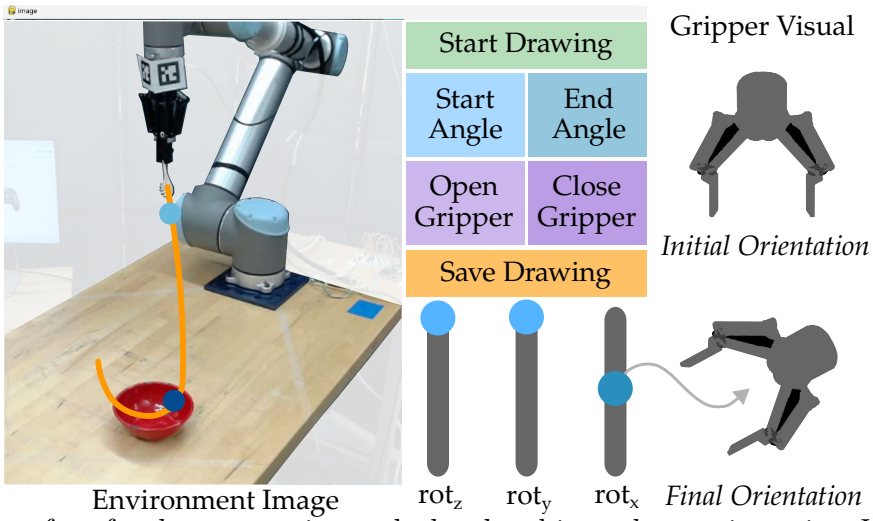


Figure 6.3: Interface for demonstrating tasks by sketching robot trajectories. We present this interface to users on a touch-screen device. Users begin by drawing a line starting from the end-effector of the robot on the environment image shown on the left. This line represents the trajectory that the robot's end-effector will follow during the task. Users then specify how the end-effector should rotate by first selecting a point on the line and then selecting the orientation at that point using the rot_x, rot_y, and rot_z sliders. We provide a visualization of the gripper orientation to help users identify their desired angles. In the same way, users can specify when the gripper should open or close by selecting a point on their line and then choosing the appropriate button.

6.3.1 Obtaining Diverse Drawings

We first present our interface design and explain how users can draw on our interface to teach the robot. To learn new tasks using imitation learning, the robot needs a dataset of state-action pairs (s, a) covering a variety of task configurations. Instead of asking users to provide these demonstrations in the real world, we use the interface shown in Figure 6.3 to obtain as much of this information as possible from trajectories drawn on images of the task.

Sketching Interface. Our interface consists of three parts, each designed to convey a specific aspect of the robot's state s_R. The first part displays an image of the robot and the environment. Users draw on this image to indicate the path the end-effector should follow to perform the task. Each point p on this line maps to some 3D end-effector position p_R. The

second part features three sliders for changing the orientation r of the end-effector about the robot's axes. Because our images are static, it can be difficult for users to imagine how the robot rotates as they move these sliders. To make the interface more intuitive, we provide a 3D visualization of the end-effector that rotates in real-time with slider input. The final part of the interface includes two buttons that open and close the robot's gripper g.

Overall, each sketch specifies a trajectory ξ_P. We update our definition of ξ_P from Section 6.2 to include the end-effector rotation $r \in \mathbb{R}^3$ about the robot's axes and a binary gripper state $g \in \{0, 1\}$ with the 2D pixel points p at each step of the trajectory. Put together, $\xi_P = [(p_1, r_1, g_1), (p_2, r_2, g_2), \cdots, (p_n, r_n, g_n)]$.

With this interface, users can demonstrate different manipulation tasks. For example, in our motivating scenario of scooping cereal from a bowl, users will draw a path from a spoon held in the robot's end-effector to the center of the bowl. Then they will select points on this path where they want to rotate the end-effector and use the sliders to specify the starting and ending orientations. Our interface then linearly interpolates between these angles to capture the scooping motion. Similarly, for the task of picking a cube and dropping it into a basket (see Figure 6.4), users will draw a trajectory from the robot's gripper to the cube and then to the basket, and then select points where they want to close and open the gripper.

Diverse Images. Our interface allows users to demonstrate the task without performing it in the real world. But to learn the task effectively, the robot needs demonstrations in various task configurations (e.g., different bowl positions in the scooping task). Rather than having users physically change the environment, we leverage existing vision-language models to digitally alter the images and create new scenarios as follows.

When the robot captures an initial image of the environment, the interface asks users to specify the relevant objects through language prompts. We feed the object prompts to a

vocabulary-based object detector, *Detic* [157] that extracts object locations o in the image, and segments out the object masks. We use these masks to generate new environment images by changing their position in the image and inpainting the area where the objects were moved from (see Figure 6.4).

In our experiments, we found that moving objects to random image locations introduces sufficient variety for learning the task. For each generated image, users provide a drawing as described earlier. These drawings are related because they show the same task, but each drawing is unique since the way that task is performed changes. This way, the robot can efficiently obtain a diverse dataset of m trajectories $\mathcal{D}_P = \{\xi_{P_1}, \xi_{P_2}, \ldots, \xi_{P_m}\}$ without requiring users to physically interact with the robot or environment.

6.3.2 Converting Drawings to Robot Trajectories

Now that we have a dataset of drawings $\mathcal{D}_P$, we need to translate these drawings into state-action pairs that the robot can use to learn a control policy. We know that each drawing ξ_P corresponds to a real-world robot trajectory ξ that the user has in mind. However, when the user projects the trajectory from a high-dimensional robot state space into a low-dimensional image space, we inevitably lose some information. In this section, we theoretically quantify this information loss, and propose a solution for mapping the 2D pixel points back to the 3D world with minimal reconstruction error.

Information Loss. We start by formalizing the fundamental gap between drawings and real robot states. In our motivating example, the robot state s_R includes the end-effector position p_R, orientation r, and gripper state g. Our drawings provide direct rotation r and gripper g information, but when we take an image of the environment, the 3D robot positions $p_R = [x_R, y_R, z_R]$ are reduced to 2D pixel points $p = [x_p, y_p]$ on the image plane. To model

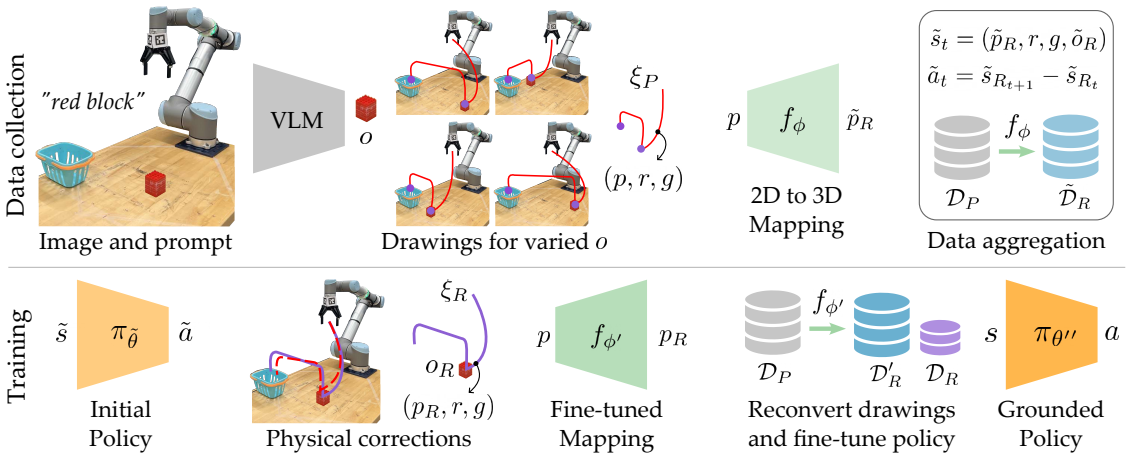


Figure 6.4: Proposed approach for Learning from 2D Drawings (L2D2). The top row outlines our procedure for collecting diverse sketching data. Our approach takes an initial image of the environment as input and creates multiple synthetic images covering a variety of task configurations. We achieve this by detecting relevant objects mentioned by the user using vision-language models (VLMs) and then randomly repositioning those objects in the scene. Users draw on these images to convey the desired task using our interface in Figure 6.3. We then use a task-agnostic mapping to convert the 2D points in each sketch to 3D positions in the real world. This ultimately results in a dataset $\tilde{\mathcal{D}}_R$ of state-action pairs $(\tilde{s}, \tilde{a})$ reconstructed from the drawings $\xi_P \in \mathcal{D}_P$. The bottom row outlines our training process. We first train a policy on the reconstructed data using behavior cloning and roll out this policy in the environment. If this policy makes any errors, users physically correct the robot's motion. These corrections result in a small dataset $\mathcal{D}_R$ of accurate physical demonstrations. We first use this physical data to refine our 2D-to-3D mapping and improve the quality of demonstrations reconstructed from the sketches. Then we leverage both these datasets to fine-tune the robot's policy and ground the robot's actions in the real world. Together, the diverse set of sketches and a few precise physical demonstrations result in an accurate and generalizable robot policy.

this projection, we first map the states to the camera frame C:

$$[p_C \;\; 1]^T = T_{CR} \cdot [p_R \;\; 1]^T \tag{6.2}$$

Here T_{CR} is a homogeneous transformation from the robot's reference frame R to the camera frame C and p_C is the position of the robot in the camera frame. Then, we project the states $p_C = [x_C, y_C, z_C]$ onto the image by dropping the z coordinate, and scaling the $x-y$

coordinates based on the dropped z_C (which is their depth) and the camera's focal lengths (f_x, f_y) to account for perspective [159].

$$p = \left[x_C \frac{f_x}{z_C},\ y_C \frac{f_y}{z_C} \right] \tag{6.3}$$

The transformation and scaling steps preserve all information — we can retrieve the robot states s_R given their distance z_C along the camera's z-axis. Therefore, we only lose information when we discard z_C to project the states onto the image. Note that a depth camera would not be helpful in this case, since users still provide a drawing in 2D. For example, the user can show how the robot moves left or right in the image plane, but not how far *into the frame* the robot should go.

We quantify this loss as the proportion of the total variance explained by the z-coordinate of all robot states $p_C \in P_C$ expressed in the camera frame [160]:

$$I_{loss}(C) = \frac{Var(z_C)}{Var(x_C) + Var(y_C) + Var(z_C)} \tag{6.4}$$

The more the robot states vary along the direction that the camera faces, the higher the information loss when representing these states with a 2D sketch.

Camera Placement. We want to minimize this information loss so that the robot can accurately reconstruct the states from the points drawn by users on the camera image. Prior approaches address this gap by obtaining additional information with each drawing in the form of distance inputs [30] (i.e., users need to manually specify the depth or height) or complementary sketches on images taken from orthogonal viewpoints [29] (i.e., users draw the same task from multiple perspectives), both of which increase the user's burden. Instead of seeking extra inputs, we propose leveraging our analysis in Equation (6.4) to find a new

camera placement that minimizes the variance along the z-axis of the image plane:

$$C^* = \arg\min_C \ I_{loss}(C) \tag{6.5}$$

We can derive the optimal solution for the above objective using Principal Component Analysis (PCA) [160]. In PCA, we compute the directions of maximum variance (i.e., the principal components) as the eigenvectors of the covariance matrix Σ_C of robot states.

$$\Sigma_C = \frac{P_C P_C^T}{n-1} \tag{6.6}$$

Here P_C is a matrix of uniformly sampled robot states represented in the camera frame. Let $\Lambda = [\lambda_1, \lambda_2, \lambda_3]$ be the eigenvalues of the covariance matrix arranged in decreasing order of magnitude and $V = [v_1, v_2, v_3]$ be an orthonormal matrix of corresponding eigenvectors. The first principal component v_1 represents the direction of maximum variance in robot states, while the last component v_3 captures the least variance. To minimize the reconstruction error, we want the z-axis of the camera to be aligned with the last eigenvector and the $x-y$ image plane to be parallel to the plane formed by the first two principal components. This arrangement will result in the least information loss I^*_{loss} given by:

$$I_{loss}(C^*) = \frac{\lambda_3}{\sum_{i=1}^{3} \lambda_i} \tag{6.7}$$

Our analysis thus far quantifies the data loss for projecting 3D robot states to 2D pixel points and proposes mitigating this loss through an improved camera setup. We do not assume any prior knowledge of the tasks that the human wants to teach, and consider the entire robot state space in our calculations. However, when performing tasks in the real world, the robot may operate in a more restricted subset of states $\mathcal{W} \subset P_C$. For instance,

the robot's workspace may be confined by an adjacent wall or be limited to a table that it is mounted on. In practice, we leverage knowledge of this *working region* $\mathcal{W}$ to compute a more domain-specific covariance matrix using $\mathcal{W}$ instead of P_C in Equation (6.6) and further improve the camera position.

With the camera held fixed, we now shift our focus to how the robot can accurately map the points p drawn on the image to the corresponding robot positions p_R.

2D to 3D Mapping. According to our camera model, the 3D robot positions are linearly projected on the 2D image plane, followed by a nonlinear scaling. When retrieving the positions from the pixel points, we can combine all inverse transformations into a single mapping $f : p \rightarrow p_R$. Prior work has shown that non-linear mappings can encode and extract information more effectively than PCA projections [161, 162, 163, 164]. Building on this insight, we model f as a non-linear function represented by a neural network $f_\phi(p)$ with parameters ϕ. We train this network on a new calibration dataset $\mathcal{D}_{map} = \{(p_1, p_{R_1}), (p_2, p_{R_2}), \ldots\}$, where each robot position p_{R_i} is uniformly sampled from the space $\mathcal{W}$ and projected onto the image using Equation (6.2) and Equation (6.3) to get the matching pixel point p_i. Note that this data is *task-agnostic* and is automatically generated before collecting the task-specific user drawings. That is, the data, while being specific to the robot's workspace, does not contain information about any specific task trajectory, gripper actuation or orientations of the robot.

We minimize the following reconstruction loss function to learn the mapping from 2D points to 3D robot states using $\mathcal{D}_{map}$:

$$\mathcal{L}_{map}(\phi) = \sum_{(p,p_R)\in\mathcal{D}_{map}} || \, f_\phi(p) - p_R \, ||^2 \tag{6.8}$$

We expect the mapping f_ϕ trained on the $\mathcal{L}_{map}$ to find non-linear manifolds that better fit the state distribution than linear principal components, leading to more accurate reconstructions.

Algorithm 4 2D to 3D Mapping

Given: Workspace $\mathcal{W}$, Camera positions $\mathcal{C}$
Find camera placement $C^* = \min_{C \in \mathcal{C}} I_{loss}(C)$
Initialize $\mathcal{D}_{map} = \{\}$
for position $p_R \sim \mathcal{W}$ **do**
    Get pixel point p with Eq. 6.2 and Eq. 6.3
    $\mathcal{D}_{map} \cup (p, p_R)$
end for
Initialize f_ϕ with random ϕ
for (p, p_R) in $\mathcal{D}_{map}$ **do**
    Compute loss $\mathcal{L}_{map}(\phi)$ using Eq. 6.8
    Update weights ϕ to minimize $\mathcal{L}_{map}$
end for
return Reconstruction function f_ϕ

In practice, this can help us further reduce the information loss [163], such that:

$$I_{loss}^{nonlinear}(C) \leq I_{loss}(C) \tag{6.9}$$

Algorithm 4 summarizes our proposed approach for bridging the gap between 2D paths drawn on images and corresponding robot trajectories in the 3D world. In the next part of our approach, we describe how we apply the learned mapping f_ϕ to convert the sketch data $\mathcal{D}_P$ collected by our interface into real-world demonstrations that can be used to train the robot.

6.3.3 Learning from Drawings

Following our outline in Figure 6.4, we have obtained a diverse dataset of sketches $\mathcal{D}_P$ with our interface and learned a task-agnostic function $f_\phi(p)$ that maps 2D image points p to 3D robot positions p_R with minimal information loss. To train the robot policy, we now need to convert the drawings into a dataset of state-action pairs (s, a) that capture the desired

behavior. In what follows, we describe our procedure for extracting this data using the learned mapping f_ϕ, and then present our core idea of grounding the drawing data with a few real-world demonstrations to fill in any remaining gaps in information and learn robust robot policies.

Data Aggregation. We begin by aggregating the data collected by our interface. Here we apply the mapping learned in Section 6.3.2 to convert each drawing $\xi_P \in \mathcal{D}_P$ to a sequence of robot states $\tilde{\xi}_R = [(\tilde{p}_{R_1}, r_1, g_1), (\tilde{p}_{R_2}, r_2,$
$g_2), \ldots, (\tilde{p}_{R_n}, r_n, g_n)]$ by reconstructing the positions.

$$f_\phi(p) = \tilde{p}_R$$

The accent $\tilde{}$ denotes the approximate reconstruction we get when moving from a 2D to a 3D space. The tuple $(\tilde{p}_R, r, g)$ defines the robot's state $\tilde{s}_R$, but it does not include the environment state o_R. In general, the user's drawings do not provide any information about other objects in the environment; the robot only observes the initial environment image and does not see how it will evolve when rolling out the drawn trajectory. For example, in Figure 6.4, users do not sketch the cube's motion. Without this feedback, the robot will go to the basket even if it fails to grasp the cube, since it does not know that the cube should move with its gripper.

One way we can address this problem is by simulating how the object state evolves throughout the task. In our implementation, we approximate object dynamics based on the object locations o detected in the initial image of the environment (from Section 6.3.1) and the reconstructed trajectory $\tilde{\xi}_R$ using the following rule: First we leverage our learned mapping f_ϕ to convert the pixel locations o to 3D object positions $\tilde{o}_R = f_\phi(o)$. Then, to model how the objects move during the task, we change their positions along with the end-effector po-

sitions $\tilde{p}_R$ whenever the gripper g is closed within a pre-specified distance from the objects. While we apply this practical simplification, it is not central to our approach; practitioners may instead employ physics-induced image manipulation [165] or video generation from task descriptions [166] to simulate object dynamics.

Once we have the simulated object positions $\tilde{o}_R$, we combine them with the reconstructed robot state $\tilde{s}_R$ to get the complete task state $\tilde{s} = (\tilde{s}_R, \tilde{o}_R)$. To create the training dataset, we now need to know what actions the robot should take in these states to achieve the sketched task in the real world. We do that by setting the robot's actions to be the difference between consecutive robot states in the reconstructed trajectory:

$$\tilde{a}_t = \tilde{s}_{R_{t+1}} - \tilde{s}_{R_t}$$

With this final piece of information, we can construct the training dataset $\tilde{\mathcal{D}}_R = \{(\tilde{s}_1, \tilde{a}_1), (\tilde{s}_2, \tilde{a}_2), \ldots\}$. Despite our efforts in accurately reconstructing the robot state and simulating object dynamics, this data can be imperfect due to the inherent lack of information in low-dimensional sketches and static images. That said, while this data lacks in accuracy, it compensates for it with a rich variety of task configurations.

In the final part of our approach, we present our idea for improving the accuracy of this diverse dataset and training a robot policy that can account for dynamic interactions that are absent in our drawings.

Grounding with Physical Demonstrations. We use the reconstructed data to train a preliminary robot policy $\pi_\theta(s) \rightarrow a$ that maps task states to robot actions. Let $\pi_{\tilde{\theta}}$ be the policy trained on $\tilde{\mathcal{D}}_R$ using the behavior cloning objective in Equation 6.1. In tasks involving simple physical interactions, this initial policy alone may be sufficient for successfully completing the task.

However, when learning contact-rich tasks, we enable humans to refine the robot's behavior with a small dataset of accurate real-world demonstrations $\mathcal{D}_R = \{(s_1, a_1), (s_2, a_2), \cdots\}$. This data can be collected by executing the task according to $\pi_{\tilde{\theta}}$ and asking users to correct the robot's trajectory, or by obtaining new demonstrations through user teleoperation. While $\mathcal{D}_R$ may not cover as many scenarios as in $\tilde{\mathcal{D}}_R$, it precisely grounds the synthetic observations in the real world. Our insight is that these data sources are complementary. Below, we describe how we can best utilize both datasets to train an accurate and generalizable policy.

Our approach does not simply merge the two sets like DAgger [16]. Rather, we utilize the real-world demonstrations to ground the robot policy in two steps: (i) recalibrating the data derived from user drawings, and then (ii) refining the robot policy using both the real-world and drawing datasets. In the first step, we recall that our 2D to 3D mapping f_ϕ was trained on positions uniformly sampled from a task-agnostic space $\mathcal{W}$ (see Section 6.3.2). By contrast, $\mathcal{D}_R$ only includes states that are specific to the desired task, which may or may not be represented in $\mathcal{W}$. To address this mismatch, we adapt our mapping to the user's task by creating new training pairs (p, p_R) with real robot positions in $\mathcal{D}_R$. With these task-specific pairs, we fine-tune our pre-trained mapping f_ϕ. Specifically, we update its weights ϕ with gradients from the loss in Equation (6.8) computed over the new states:

$$\phi' \leftarrow \phi - \alpha \nabla_\phi \mathcal{L}_{map} \quad \text{using } (p, p_R) \sim \mathcal{D}_R \tag{6.10}$$

The updated mapping $f_{\phi'}$ captures task-specific information and should thus provide a more precise 2D to 3D mapping for the task at hand. Now that we have this improved mapping, we reapply it to the original user drawings $\mathcal{D}_P$ to construct a more accurate dataset of reconstructed trajectories $\mathcal{D}'_R$.

In the second step, we leverage this new dataset of demonstrations $\mathcal{D}'_R$ extracted from draw-

ings and the physical demonstration data $\mathcal{D}_R$ to retrain the robot policy. Here we highlight that besides task-specific state information, $\mathcal{D}_R$ also contains rich object interactions. For instance, users may demonstrate grasping the cube multiple times if they fail during initial tries. These interactions offer valuable feedback that can help the robot adapt to real-world outcomes, e.g., learning not to lift the cube until it is successfully grasped.

To incorporate this knowledge in training the robot's policy, one may simply try to combine the two datasets. But in practice, due to the imbalance between the sizes of the real-world and drawing data, the learned policy becomes biased towards the trajectories in $\mathcal{D}'_R$ and ignores the contact-rich information in $\mathcal{D}_R$ [167]. In our implementation, we mitigate this bias by sequentially training the robot's policy on each dataset using the standard behavior cloning loss from Equation (6.1):

$$\begin{aligned} \theta' &\leftarrow \theta - \beta \nabla_\theta \mathcal{L}_{BC} \qquad \text{using } (s', a') \in \mathcal{D}'_R \\ \theta'' &\leftarrow \theta' - \beta \nabla_{\theta'} \mathcal{L}_{BC} \qquad \text{using } (s, a) \in \mathcal{D}_R \end{aligned} \tag{6.11}$$

Here θ' denotes the intermediate policy parameters after training on $\mathcal{D}'_R$. These parameters capture behaviors across various task settings. We then fine-tune θ' on $\mathcal{D}_R$ to obtain the final grounded policy $\pi_{\theta''}$. This final training phase helps the robot extrapolate the real-world knowledge to the diverse settings illustrated in our drawings.

Overall, our training procedure enables the robot to not only adapt to different task configurations — because of the diverse drawings collected by our interface — but also realize the physical consequences of its actions and compensate for any inaccuracies in 3D reconstruction and simulation of object dynamics.

Algorithm 5 *L2D2*

Given: Drawings $\mathcal{D}_P$, Mapping f_ϕ

Initialize $\tilde{\mathcal{D}}_R = \{\}$
function $LearnFromDraw(\mathcal{D}_P, f_\phi)$
    for ξ_P in $\mathcal{D}_P$ **do**
        // *Data Aggregation from Drawings*
        for (p_t, r_t, g_t) and p_{t+1} in ξ_P **do**
            Convert $\tilde{p}_{R_t} \leftarrow f_\phi(p_t)$
            Convert $\tilde{p}_{R_{t+1}} \leftarrow f_\phi(p_{t+1})$
            Robot state $\tilde{s}_R = (\tilde{p}_R, r, g)$
            Compute action $\tilde{a}_t = \tilde{s}_{R_{t+1}} - \tilde{s}_{R_t}$
            Get object state $\tilde{o}_R$ from image
            $\tilde{\mathcal{D}}_R \cup (\tilde{s}_R, \tilde{o}_R, \tilde{a}_t)$
        end for
    end for

    Initialize π_θ with random θ
    for (s, a) in $\tilde{\mathcal{D}}_R$ **do**
        Compute loss $\mathcal{L}_{BC}(\theta)$ using Eq. 6.1
        Update weights θ to minimize $\mathcal{L}_{BC}$
    end for

    return Robot policy π_θ
end function

// *Learning initial robot policy from drawings*
$\pi_{\tilde{\theta}} \leftarrow LearnFromDraw(\mathcal{D}_P, f_\phi)$

Collect real-world demonstrations $\mathcal{D}_R$

// *Fine-tuning the reconstruction function*
for (p, p_R) in $\mathcal{D}_R$ **do**
    Compute loss $\mathcal{L}_{map}(\phi)$ using Eq. 6.8
    Update weights ϕ to minimize $\mathcal{L}_{map}$
end for
Task-specific mapping $f_{\phi'}$

// *Learning intermediate policy parameters*
$\pi_{\theta'} \leftarrow LearnFromDraw(\mathcal{D}_P, f_{\phi'})$

// *Fine-tuning robot policy*
for (s, a) in $\mathcal{D}_R$ **do**
    Compute loss $\mathcal{L}_{BC}(\theta')$ using Eq. 6.1
    Update weights θ' to minimize $\mathcal{L}_{BC}$
end for

return Grounded robot policy $\pi_{\theta''}$

6.3.4 Algorithm Summary

Our proposed approach for **L**earning from **2D D**rawings: L2D2, is summarized in Algorithms 4 and 5. Our code is available here: https://github.com/VT-Collab/L2D2

We first apply PCA to find an optimal camera placement C^* for taking images of the environment, and learn a 2D to 3D mapping f_ϕ following the lines 8−12 in Algorithm 4. This mapping reconstructs the real-world positions corresponding to the points in images taken from C^*. We then leverage vision-language models [157] to synthetically generate images of varying task configurations and ask users to convey their desired task by drawing on these images using our interface in Figure 6.3. This creates a diverse dataset of drawings $\mathcal{D}_P$.

To learn from these drawings, we use our mapping f_ϕ to create a dataset of reconstructed state-action pairs $(\tilde{s}, \tilde{a}) \in \tilde{\mathcal{D}}_R$ as outlined in Algorithm 5. With this data, we train an initial robot policy $\pi_{\tilde{\theta}}$. Since this policy is learned from static drawings, it may fail to capture dynamic real-world interactions. We bridge this gap by grounding the policy with a few real-world demonstrations $\mathcal{D}_R$. This process involves two steps: First, we improve our mapping $f_{\phi'}$ using task-specific states from the real-world data (lines 26−30) and reconstruct the drawing demonstrations $\mathcal{D}'_R$. Second, we leverage both $\mathcal{D}'_R$ and $\mathcal{D}_R$ to sequentially train a grounded robot policy $\pi_{\theta''}$ (lines 32−38). The diverse drawings enable the robot to generalize across various task configurations, while the real-world data teaches it to account for environment dynamics.

Although ours is not the first approach for learning from sketches, it introduces two key advances: synthetically generating diverse drawings and grounding them with real-world demonstrations. In the following section, we evaluate how these advances enable humans to teach robots more efficiently and accurately than prior sketch-based learning approaches.

6.4 Real-World Experiments

In Section 6.3, we presented L2D2, an interface-based approach for teaching robots with task sketches. We now evaluate our proposed approach through real-world experiments. Specifically, we compare L2D2 to state-of-the-art methods for learning from drawings, as well as standard approaches for learning from physical demonstrations. We break down our experiments into three parts. First in Section 6.4.1, we evaluate the performance of L2D2 on short manipulation tasks with data provided by expert users. Next, in Section 6.4.2, we conduct an in-person study to assess whether novice users can leverage our approach to teach robots efficiently. Finally, in Section 6.4.3, we test if expert users can apply our approach for teaching robots to perform longer manipulation tasks.

Independent Variables. We compare L2D2 to three state-of-the-art baselines that leverage different feedback mechanisms for teaching manipulation tasks to a robot arm: using teleoperated demonstrations (**Teleop**), using sketches to condition the robot policy (**RT-Traj**) [30], and using sketches in two camera images to reconstruct the 3D demonstrations (**S2S**) [29]. We also compare against two ablations of our proposed approach. In the first ablation, we evaluate the performance of L2D2 when it only has access to drawings collected using our interface (**L2D2-D**). In the second ablation, we evaluate the performance of our approach when it is trained with just the small set of physical corrections that we collect to ground the drawings (**Teleop-min**). Below, we describe these approaches and their training and operating procedures in detail:

- **Teleop:** In Teleop, users directly control the position and orientation of the robot's end-effector, and actuate the robot's gripper using a joystick. With this approach, users can provide physical demonstrations that accurately capture their desired behavior in the real-world environment.

- **RT-Traj:** In this approach, users draw the robot's trajectory on a camera image. Unlike L2D2, RT-Traj does not get rotational inputs, instead, it asks users to specify the real-world heights for key points along the sketched trajectory. This sketch is used to condition a pre-trained transformer that outputs robot actions. So for each sketch, we roll out these actions to record the corresponding physical demonstration (that we use to train the robot policy).

  The transformer model used to map the 2D sketches to the 3D world is trained on large multi-task data. To reduce this data requirement, we simplify its implementation by directly giving it the sketched trajectories rather than images of the sketch as in the original paper [30]. In our experiments, we collect 160 task-specific demonstrations to train this mapping. These demonstrations are different from those used to train the robot policy.

- **S2S:** This is another sketching approach that takes images of the environment, but from two orthogonally placed cameras instead of one. Users draw on both camera images to convey a single corresponding real-world demonstration. So users need to imagine how the same robot trajectory would appear from two different viewpoints. Similar to RT-Traj, S2S does not obtain rotational inputs from users and requires a pre-trained autoencoder network to convert the paired 2D sketches into one real-world trajectory. We use the same 160 task-specific demonstrations that we collected for RT-Traj to train the mapping for S2S following the procedure in [29]. However, unlike RT-Traj, S2S directly maps the sketches to real-world demonstrations without the need for rolling out the sketches in the real world.

For all baseline methods, users need to physically interact with the environment to vary the task scenarios, which they do not need to do with L2D2. We also do not require large task-specific datasets to learn a 2D to 3D mapping. Instead, we use a task-agnostic mapping

f_ϕ and a few real-world demonstrations to ground the diverse drawings collected by our interface.

Experimental Setup. In all our experiments, we use a 6-DoF Universal Robots UR-10 robot arm equipped with a two-finger Robotiq gripper. All the tasks are performed on the table on which the robot arm is mounted. We leverage the procedure in Section 6.3.1 to position the camera as best as we can to minimize the information loss when learning from 2D drawings. The same camera position is used to collect drawings for all sketch-based methods and all the tasks in the experiments. In addition to this camera, we place a second camera almost orthogonal to the first one for the S2S baseline.

During each interaction, the user first resets the environment by physically changing the positions of task-relevant objects. This happens automatically before each drawing for L2D2 and L2D2-D. Then they provide a sketch or a teleoperated demonstration in the current scenario to convey their task to the robot.

6.4.1 Short Horizon Tasks

We start by evaluating the performance of all methods in short-horizon manipulation tasks like lifting or pushing objects placed on a table. Here we provide the same amount of expert demonstrations (sketched or physical) to each approach and test how well the resulting policy performs the demonstrated task.

Task Descriptions. In this experiment, expert users taught two tasks to the robot arm: *Lift* and *Push* (shown in Figure 6.5). In the *Lift* task, the expert's goal was to teach the robot to reach a red cube placed on the table, close its gripper, and lift the object above a specified height threshold. In the *Push* task, the expert had to teach the robot to reach for a bowl and move it to the center of the table. For the *Lift* task, the experts were instructed

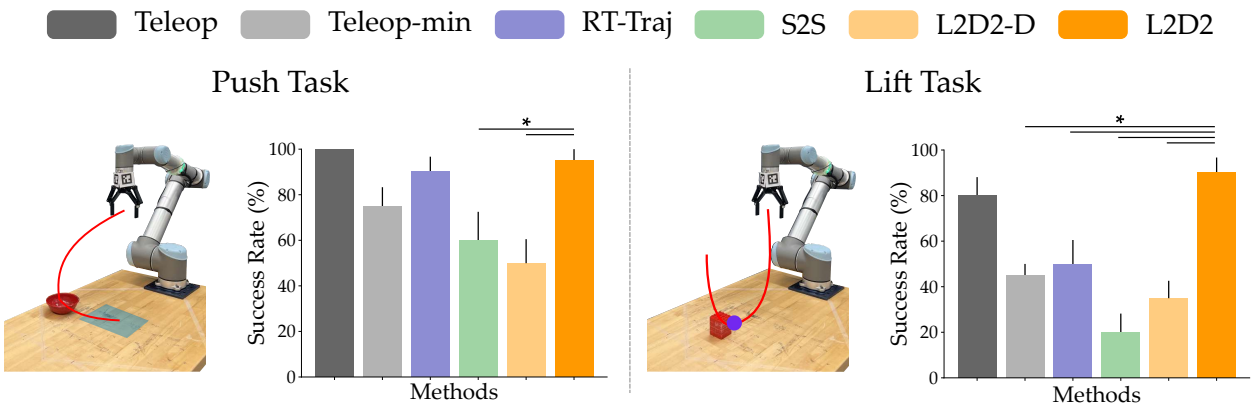


Figure 6.5: Results for short-horizon tasks with expert data. (Left) The user is trying to teach the robot to push the bowl to the center of the table (blue region) by drawing the trajectory for the task on the image of the environment. (Right) The robot is learning to pick up a block from the drawings provided by the user. We report the success rate for both tasks averaged over 10 independent rollouts with varying object locations. The error bars show the standard error around the mean (SEM), and $*$ denotes statistical significance ($p < 0.05$). For the push task, L2D2 achieves a higher success rate than L2D2-D and S2S, while for the *Lift* task L2D2 performs similar to Teleop and outperforms all other baselines.

to randomly vary the cube's position to uniformly cover the entire table, and for the *Push* task, they were asked to ensure that the random bowl placement was away from the table's center.

Data Collection and Training. We fix the total number of demonstrations that the experts provide for each method. Specifically, they provided a total of 60 demonstrations to the robot per task. When using Teleop, the experts gave 60 physical demonstrations. For RT-Traj, they drew 60 sketches, while for S2S, they made 120 drawings (one in each camera frame per demonstration). Finally, for L2D2, the experts provided 50 drawings and 10 physical demonstrations.

Once we have the data for all methods, we use the same network architecture to train their respective policies. Specifically, we use a multi-layer perceptron (MLP) that takes the end-effector state and object features as input and outputs the end-effector velocities.

Dependent Variables. We evaluate the performance of each approach in terms of success rate for completing 10 different instances (i.e., initial object positions) of each task. In each instance, we compute success by breaking down the task into smaller segments and measuring the fraction of segments that were completed successfully. For the *Lift* task, we divide the task into reaching and lifting segments to measure success. For example, if the robot reaches the block but fails to grasp and lift it, we have 50% success. Whereas if the robot successfully grasps and lifts the block, the success is 100%. Similarly, we divide the *Push* task into two segments: reaching the bowl and pushing it to the center, each accounting for 50% of the task success. In addition to the success rate for the tasks, we also measure the time that the experts spend in providing drawings and physical demonstrations to the robot.

Results. Figure 6.5 summarizes our results averaged over the 10 testing scenarios.

We first analyze the success rates for the *Push* task. A One-way ANOVA revealed that the teaching method had a significant effect on task success ($F(5, 54) = 6.02$ and $p < 0.05$). Post-hoc comparisons indicated that L2D2 significantly outperformed S2S ($p < 0.05$) while being as successful as Teleop ($p = 0.67$) and RT-Traj ($p = 0.67$). We found similar results for the *Lift* task. A One-Way ANOVA revealed that the choice of the teaching method had a significant effect on the robot's performance ($F(5, 54) = 11.55$ and $p < 0.05$). Post-hoc analysis indicated that L2D2 significantly outperformed both S2S and RT-Traj baselines ($p < 0.05$), while performing similarly to Teleop ($p = 0.37$). Additionally, during the experiments we observe that when demonstrating the task using drawings the experts only spent $\sim 50\%$ of the time that they spent in teleoperating the robot.

This result shows that, while enabling faster data collection and using only 10 real-world demonstrations, L2D2 was as performant as Teleop, which has access to 60 physical demonstrations. In contrast, S2S failed to reach the block accurately in many instances. We

hypothesize that this is due to the difficulty of visualizing how the robot's trajectory would appear in two different camera frames. Our results for S2S match those reported in [29] for the tasks involving gripper actuation.

When comparing to ablations of our approach, we found that L2D2 achieves a significantly higher success rate than L2D2-D in both tasks ($p < 0.05$). It also outperforms Teleop-min in the *Lift* task ($p < 0.05$) where there is a greater variation in object positions. This highlights the benefit of grounding the drawings with a few physical demonstrations, while also showing how the diverse drawings can help the robot perform better under varying task configurations.

6.4.2 User Study

In the previous section, we demonstrated that our proposed approach can learn short manipulation tasks effectively with drawings from expert users. These users were experienced in using the sketching interfaces and were thus able to provide high-quality drawings to the robot. However, can novice users learn to use our interface effectively, and will they prefer our proposed approach for teaching robots? Along with improving performance, it is equally important for a teaching interface to be intuitive for human teachers. Hence, in this section, we conduct a user study with 12 participants who had never used the drawing interfaces and assess whether it is easy for these end-users to teach the robot by drawing on our interface. We evaluate the time they need to provide drawings and the resulting performance of robot policies learned from their data.

Task Descriptions. In this study, the users were tasked with teaching two tasks: *Pick and Place* and *Scooping*. In the *Pick and Place* task, the users' goal was to teach the robot to pick up a block from different locations in the environment and drop it inside a bin kept at

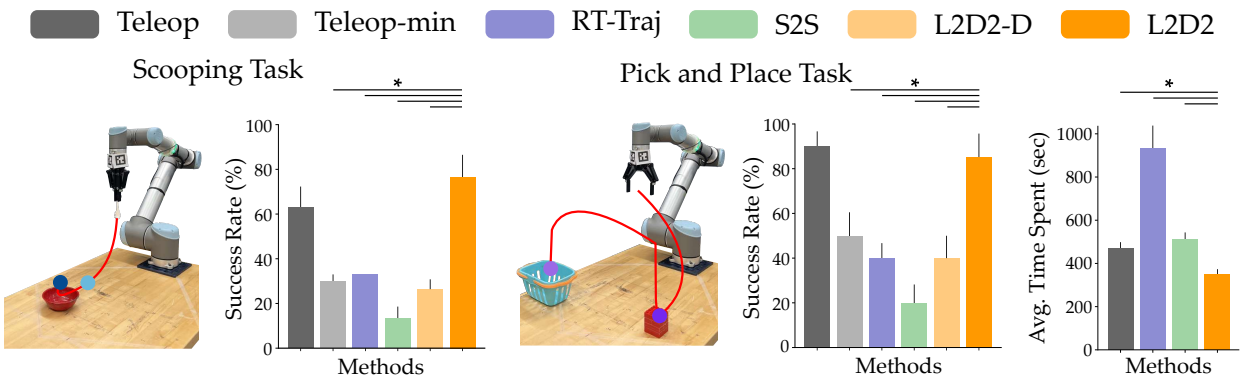


Figure 6.6: Objective results for the user study in Section 6.4.2. Participants teach the robot to perform two tasks in the environment: *Scooping* and *Pick and Place*. In each task, they provide physical demonstrations through teleoperation (Teleop) and drawings using our proposed approach (L2D2) and two sketching baselines, RT-Traj and S2S. We record the total time spent by the users in providing demonstrations to the robot and the average success rate of the learned policy evaluated over 10 rollouts. The error bars in the plots show the SEM and $*$ signifies that L2D2 had a significantly better performance than the baseline. Across both tasks, users spend significantly less time providing demonstrations using L2D2, and achieved a significantly higher success rate as compared to RT-Traj and S2S.

a fixed location. For the *Scooping* task, the robot started with a spoon in its gripper. The users' goal was to teach the robot to reach inside the bowl and rotate the robot's gripper such that the spoon scoops the contents of the bowl. For both tasks, users were asked to randomly place the objects in the environment at the start of each interaction, except when using L2D2.

Participants and Procedure. We recruited 12 participants (3 female, average age 24.7 ± 5.1) from the Virginia Tech community. 10 out of the 12 participants in the user study had prior experience in working with robots. However, none of the users had previous experience of using drawing interfaces to demonstrate the tasks to the robot. Participants received monetary compensation for their time and gave written consent prior to the start of the experiment under Virginia Tech IRB #23-1237.

The participants provided demonstrations with each method for both tasks. We counter-balanced the order in which the methods were presented to the participants using a Latin

Square design (e.g., three participants started with L2D2, three started with Teleop, etc.). Before providing demonstrations with each method, the participants were given 5 minutes to practice using the joystick or the sketching interface to teach the robot. Once the participants were familiar with the interface, they provided a total of 5 demonstrations per task. For Teleop, the participants gave 5 teleoperated demonstrations. For RT-Traj and S2S, they made 5 and 10 drawings, respectively, while with L2D2, they provided 4 drawings and 1 physical demonstration. After using each method, participants answered a survey to report their subjective experience of demonstrating both tasks with that interface.

We combine the demonstrations provided by all users to create the training datasets for each method.

Dependent Variables. Similar to Section 6.4.2, we measure the success rate across 10 task configurations with varying object positions to evaluate the performance of the learned policies. For the *Scooping* task, we compute success by breaking the task into two segments: reaching into the bowl and performing a scooping motion. Each segment accounts for 50% of the task success. Likewise, for the *Pick and Place* task, we create three segments: reaching for the block, grasping and lifting it, and carrying it to the bin, each contributing to a third of the task success. For example, if the robot managed to pick the block but failed to take it to the bin, the policy rollout would be 66.6% successful.

In addition to the success rate, we also measure the total time taken by each participant to demonstrate both tasks with each method, and their subjective responses to a 7-point Likert Scale survey. The survey questions were arranged into six multi-item scales: how *easy* it was to provide the demonstrations, how *intuitive* the demonstration interface was, and how much *effort* was required to teach the robot. At the end of the study, participants were asked two forced-choice questions: whether they preferred teaching the robot through drawings or teleoperation, and which of the three sketching approaches they preferred to use.

Hypotheses. We had the following hypotheses:

H1. *L2D2 will perform similarly to Teleop, but will outperform other baselines across both tasks.*

H2. *The users will require less time to demonstrate the tasks using L2D2 as compared to the baselines.*

H3. *Users will perceive that drawings require less effort as compared to physical demonstrations, and will prefer using L2D2 over other drawing approaches.*

Results. The results for this user study are summarized in Figure 6.6 and Figure 6.7.

We first evaluate the performance of each method using the data from inexperienced users. A One-way ANOVA revealed that the teaching method had a significant effect on the success rate for the *Scooping* ($F(5,$
$54) = 9.54, p < 0.05$) and *Pick and Place* ($F(5, 54) = 14.22, p < 0.05$) tasks. Post-hoc comparisons for *Scooping* showed that L2D2 outperforms RT-Traj and S2S baselines ($p < 0.05$), but performs similarly to Teleop ($p = 0.69$). Likewise, for the *Pick and Place* task, L2D2 achieves a similar success rate to Teleop ($p = 0.14$) and outperforms all other baselines ($p < 0.05$), providing support for our hypothesis **H1**. We also observe that Teleop-min and L2D2-D have a similar performance across both the tasks. (*Scooping*: $p = 0.433$ and *Pick and Place*: $p = 0.717$). This indicates that only drawings or only a few physical demonstrations were insufficient to accurately convey these tasks to the robot. However, leveraging L2D2 that combines both of these teaching mechanisms, the end-users in our user study were able to effectively teach the tasks to the robot.

While L2D2 performs on par with Teleop, its benefits become apparent when we compare the total time taken by users to provide the same amount of data with each approach. A repeated measures ANOVA with Greenhouse-Geisser correction revealed that the methods had a significant effect on the overall time spent in demonstrating the task

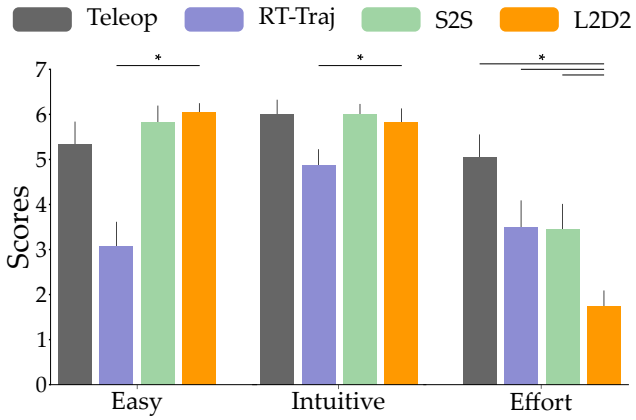


Figure 6.7: Subjective results from our user study. Higher ratings for *Easy* and *Intuitive* scales represent better subjective experience. On the other hand, for the *Effort* scale, a lower rating indicates that the user had to spend less effort in demonstrating the tasks to the robot. The participants perceived our approach (L2D2) to be as easy and intuitive as teleoperation, and indicated that it requires significantly less effort than all baselines. The error bars show the SEM, and $*$ denotes statistical significance ($p < 0.05$).

($F(1.421, 15.63) = 119.99, p < 0.05$). Post hoc analysis also indicated that users spent significantly less time demonstrating the tasks using L2D2 than all baselines ($p < 0.05$). On average, when providing drawings, the novice users required less than 75% of they time that they spent in providing phycical demonstrations. This result supports hypothesis **H2** and demonstrates that L2D2 achieves both learning efficiency and task performance by integrating synthetic drawings with physical feedback.

We next analyze the subjective results from the Likert scale survey summarized in Figure 6.7. After verifying that our scales were reliable (Cronbach's $\alpha > 0.7$), we grouped the responses for each scale. We performed repeated measures ANOVA for each scale with the necessary corrections for violation of sphericity. The tests indicated that the teaching method had a significant effect on the ease ($F(3, 36) = 11.14$, $p < 0.05$) and intuitiveness ($F(2.01, 24.1)$, $p < 0.05$) of providing demonstrations. Post-hoc comparisons revealed that users fou-nd L2D2 to be as easy and intuitive to use as Teleop, despite not being able to see the robot and objects move in our interface. Lastly, the test also showed a significant

effect of method on the perceived effort ($F(2.2, 26.41) = 13.61$, $p < 0.05$), with post hoc comparisons indicating that users put significantly less effort for L2D2 than the baselines ($p < 0.05$). Analyzing the users' subjective responses provided in the survey (see Appendix B.1), we believe that this is because participants had to manually change the positions of task-relevant objects for all baselines, while our approach leveraged vision-language models to automatically vary the object positions in the camera images.

Overall, 66.6% of users reported that they preferred using drawings to teach the robot as compared to using direct teleoperation, and 91.67% of users stated that they would prefer using L2D2 over other sketching approaches. These subjective results support our hypothesis **H3**. Users particularly disliked RT-Traj, stating that *"specifying the height of keypoints is not clear"* in their open-ended responses. Due to this ambiguity, users were unable to select the correct heights, resulting in rollouts where the robot failed to grasp the block or rotate the spoon. While users did not face the same problem with S2S, the interface did not allow them to specify end-effector rotation for the *Scooping* task, and their drawings for the two camera frames were often misaligned in *Pick and Place*, leading to poor reconstruction and task performance. Instead of obtaining heights or drawings from two viewpoints, our approach used a few teleoperated demonstrations to better reconstruct the robot's trajectory from sketches, which made our interface much more intuitive for end-users and resulted in more accurate learning.

6.4.3 Long Horizon Task

So far, we have demonstrated that our approach works with both expert and novice users. However, our evaluations have only included short manipulation tasks with a single object in the environment. This makes it easy to draw paths that go from the robot's end-effector to

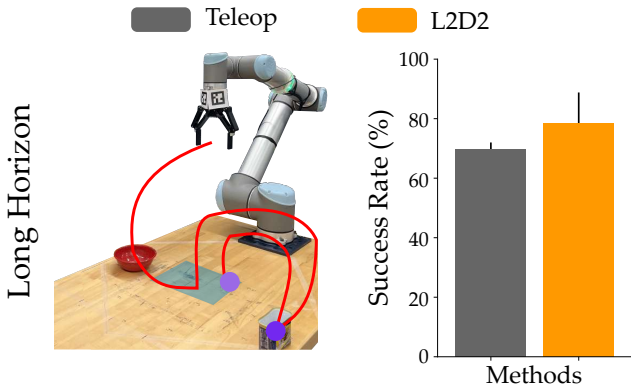


Figure 6.8: Experimental results for the long-horizon task in Section 6.4.3. The expert's goal is to teach the robot to push a bowl to the center of the table, followed by picking up a can and placing it next to the bowl. (Left) Environment setup and an example of the drawing provided by the expert. (Right) The success rate of the policies learned from demonstrations collected using Teleop and L2D2. We observe a similar performance for both approaches. Error bars show the SEM.

the object of interest. But can our approach work for *long-horizon* tasks where the robot must sequentially interact with multiple objects? In this section, we explore whether our sketch-based approach (L2D2) can be leveraged to teach these *long-horizon* tasks. Specifically, we compare the performance of L2D2 to the best-performing baseline from the previous sections (Teleop).

Task Description. The experts were tasked with teaching a *Long Horizon* task of setting up a dining table. Following recent works in long-horizon imitation learning [168, 169, 170], we formulate our long-horizon task as a sequence of shorter subtasks involving sequential interactions with two objects in the environment — an empty bowl and a can of food. In the first subtask, the robot had to move a bowl to the center of the table, similar to the *Push* task from Section 6.4.1. Then, in the second subtask, the robot had to pick up the can and place it next to the bowl. The bowl was randomly initialized on the table while the can started in a fixed location.

Data Collection. Similar to Section 6.4.1, we conduct this experiment with data collected

from expert users. For this task, the experts provided a total of 125 demonstrations to the robot. For Teleop, the experts provided a total of 125 physical demonstrations by controlling the robot's end-effector velocity using a joystick. On the other hand, for L2D2, the experts provided 100 drawings and 25 physical demonstrations to the robot.

Dependent Variables. Consistent with the previous sections, we evaluate the performance of the robot in terms of success rate across 10 independent task settings. We compute the success rate by breaking each subtask into smaller segments. We divide the first subtask of moving the bowl into two segments: reaching for the bowl and pushing it to the center of the table, where each segment accounts for 50% of the subtask success. The next subtask of shifting the can is broken into three segments: reaching for the can, grasping and lifting it, and placing it next to the bowl, each representing 33.3% of the subtask success. For example, if the robot completes the first subtask and reaches the can, but fails to close its gripper to grasp it, the rollout is considered to be 66.6% successful.

Results. The results for this experiment are summarized in Figure 6.8. An independent samples t-test did not reveal a significant difference in the success rates of policies trained with L2D2 and Teleop ($t(18) = 11.37$, $p = 0.441$). This indicates that our sketching-based approach, requiring the expert users to spend $\sim$ 50% less time in providing drawings as compared to teleoperation, is also effective in *long-horizon* tasks, achieving a performance similar to training with an equivalent dataset of accurate real-world demonstrations.

6.5 Conclusion

In this manuscript we proposed L2D2, a drawing-based interface for imitation learning. Throughout our work we recognize that teaching robot arms by sketching the desired trajectory brings advantages and disadvantages. The key advantage is the ability to rapidly

collect low-effort human demonstrations (i.e., drawings). The disadvantages stem from the fundamental gap between drawing the task in a static, 2D image and actually performing that same task in our dynamic, 3D world.

To maximize the *potential* of learning from drawings, we leveraged vision and language models to segment the initial image and generate a corpus of synthetic environments. Human teachers could rapidly draw on multiple of these images to convey diverse examples of the intended task (e.g., showing the robot arm how to grasp a block at various positions on the table). This ultimately resulted in a sketching interface that users seamlessly interacted with to draw and annotate their desired task. Participants perceived our method for teaching by drawing to be easier, more intuitive, and less taxing than state-of-the-art baselines.

To minimize the *weaknesses* that are inherent to drawn demonstrations, we developed a two-part solution. First, we derived an optimization approach based on Principal Component Analysis to place the robot's camera. By drawing on the images obtained from this camera location, L2D2 mitigated information loss between the user's 2D sketch and the robot's 3D workspace. Next, to address the gap between static images and dynamic interactions, we grounded the human's drawings by collecting a small set of physical demonstrations. Using these demonstrations L2D2 refined its understanding of what the drawings meant, and reached a policy that captured how objects should transition. Our experiments across different tasks with novice and expert users indicated that L2D2 learned a policy that was as effective as policies trained with teleoperated data, but required significantly less time for data collection as compared to physical demonstrations.

Limitations. Our results suggest that the benefits of L2D2 increase as the number of demonstrations required to teach the robot scales up: with novice users, providing 100 demonstrations takes 20 minutes less with L2D2 than with traditional teleoperation. We therefore see L2D2 as an important step towards intuitively teaching robot arms to perform

real-world tasks. One limitation with implementation of L2D2 is that it requires access to data that effectively covers and captures diverse representations of the robot's workspace to find an optimal camera location. This limitation can be addressed in most manipulation tasks if we have access to the robot's urdf and task environment to compute the robot's workspace and inaccessible regions.

Another limitation of L2D2 is that users need to translate the behavior they want the robot to perform into a drawing — and this can be challenging within long-horizon settings and with deformable objects that may need force feedback for successful task execution. For instance, if the robot needs to manipulate multiple objects, the human illustrator may be unsure what object the robot is currently holding when completing their drawing. Similarly, if two objects are dynamically coupled in the task (e.g., the robot is pushing a plate with silverware on top of that plate), the drawing may not fully capture the interactions between these objects. Our current solutions — including grounding the drawings with real-world data — help to address this challenge. However, this may not work effectively with deformable objects that may change shapes during interaction, as we would need more real-world demonstrations to fill this information gap. Future work should ensure that the visualization of the task captures object manipulation, and that users understand how the gaps in their drawings can be corrected through physical interventions. Additionally, future works can explore how VLMs can be leveraged to generate some intermediate images for the task that the users are performing. The approach could then transition between these images to make it easier for the end users to keep track of their actions in long-horizon tasks. This can also enable the algorithm to adapt to deformable objects and track their intermediate states, which may not be possible with directly simulating these objects to move with the robot gripper.

Chapter 7

Conclusion

In this thesis, we have explored different aspects of teaching robots to perform diverse tasks ranging from object manipulation to driving around humans. We focused on imitation learning algorithms and how robots can leverage these algorithms to learn from human inputs or available datasets to learn to perform a task successfully. This approach is especially critical for development of robots capable of working around human users while accounting for their preferences and adapting to previously unseen changes in the environment.

We first explore online learning scenarios where humans are available to teach and correct robot behavior. Recognizing the value of different forms of feedback in robot learning, we proposed an algorithm that unifies demonstrations, corrections, and preferences within a flexible reward learning framework. By combining passive feedback (demonstrations and corrections) with active querying (preferences), the robot achieved higher accuracy than with any single feedback type, showing that diverse forms of feedback allows humans to provide more accurate guidance, thus enabling the robots to learn the desired behavior efficiently.

Because human feedback can be noisy or biased—due to task complexity or physical constraints—we next explore online learning from suboptimal passive feedback. We formulate the robot's online learning rule as a dynamical system, and leverage Lyapunov stability analysis to derive conditions that lead to convergence. We then leverage the stability conditions to modify the robot's online learning rule to expand the basins of attractions and ensure convergence for a wider range of human actions. This approach improved robustness of online learning

rules to imperfect human feedback and enabled the robots to more accurately and rapidly infer the preferences of the users.

We then focus on scenarios where the human is not available to provide additional feedback. In such scenarios, the policies learned from offline datasets may be prone to covariate shift when they encounter data that was not seen in the demonstration dataset. We formulate covariate shift as a linearized dynamical system, derived conditions for stability, and leveraged these conditions to develop Stable-BC, an algorithm that ensures local policy stability around states seen during training. Stable-BC proved effective in diverse domains, including autonomous driving, quadrotor navigation, visual learning, and dynamic games like air hockey, with limited amount of demonstration data.

Finally, recognizing the physical effort required to provide demonstrations, in either online or offline learning settings, we propose L2D2, an algorithm which enables the novice end-users to teach the robot by providing sketches on the images of the environment. Using vision and language segmentation models, we autonomously generate synthetic environment images with varied object locations to collect diverse demonstration data. We then combine these drawings with a few real world demonstrations to ground the robot's learned policy in the real world. This approach of teaching the robots using a combination of drawings and physical demonstrations reduced the time and effort spent by the users in teaching the robot, while also enabling the robot to generalize to diverse task settings.

Overall, our research advances robot learning from humans by considering different types of feedback, their quality and availability. Collectively, these contributions form a cohesive framework that enhances stability, adaptability, and efficiency in both online and offline imitation learning settings. Together, they move us towards development of robots that integrate seamlessly into human environments, operate reliably under uncertainty, and adapt to diverse tasks with minimal user burden.

7.1 Possible Directions for Future Works

This thesis introduces approaches in the domain of offline and online imitation learning from diverse forms of human feedback to enable the robots to learn stable behaviors that are robust to different scenarios that it may encounter in the real world. However, several avenues remain open for future exploration. One important direction is the development of enhanced safety mechanisms to ensure secure interactions between humans and robots in dynamic environments. Future efforts could focus on adaptive safety constraints that do not rely on predefined rules but instead evolve with the robot's learning process.

Another promising extension of this research lies in human-robot collaboration in scenarios involving multiple users providing simultaneous or even conflicting feedback. Studying such scenarios could yield insights into group-based learning paradigms and strategies for reconciling diverse inputs. Finally, large-scale real-world deployment and longitudinal user studies are essential for validating the proposed methods across domains such as industrial automation, healthcare, and household assistance. These studies would provide critical feedback to further refine learning models, ensuring they remain robust, adaptive, and user-friendly over extended operation. By pursuing these directions, the field can advance toward fully autonomous systems capable of safe, seamless, and efficient collaboration with humans in diverse, unpredictable environments.

Appendices

Appendix A

Additional Experiments for StROL

In our simulation and real world experiments, we test our approach StROL with users with varying levels of noise and bias in their actions and in situations where the task that the users wanted to teach the robot was not included in the prior of known tasks provided to the robot. Here we perform some additional experiments to test the sensitivity of our approach to the hyperparameters and in scenarios where the users change their intended task preferences midway through the teaching process.

A.1 Effect of Relative Weight of the Corrective Term

In StROL we augment the original learning rule of the robot g with a corrective term $\hat{g}$ to expand the basins of attractions around the prior of known tasks to ensure convergence in face of noisy and biased human actions using the following equation:

$$\tilde{g} = g + \lambda \hat{g} \tag{A.1}$$

where λ is a hyperparameter that determines the relative weight of the two learning rules. In our simulation and real world experiments we set the value of λ to be 1. That is, both the original learning rule and the corrective term had the same contribution to the learning of the robot. But what would happen if we change the value of this hyperparameter and

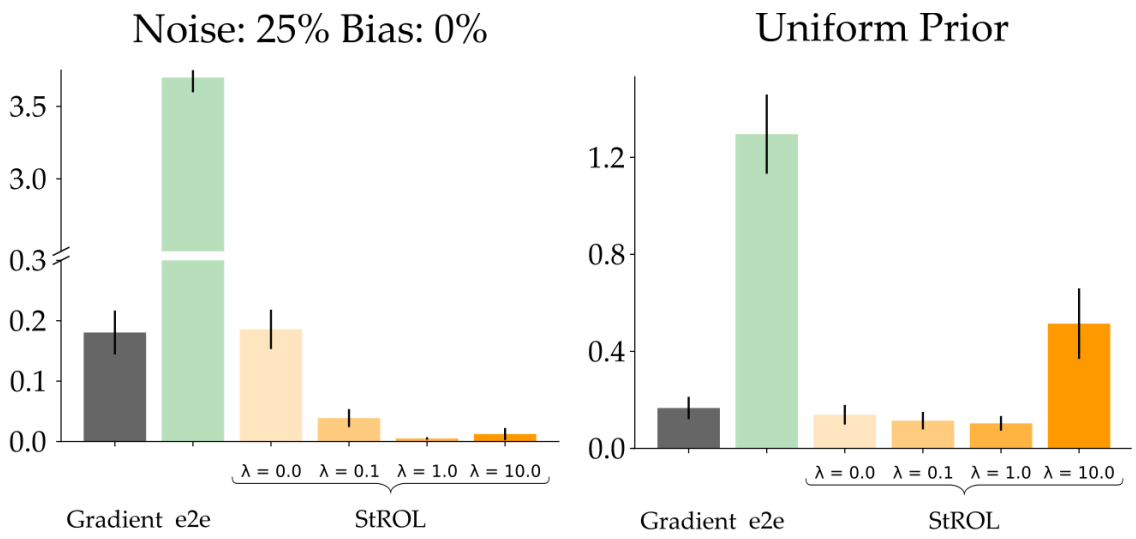


Figure A.1: Performance of StROL with varying relative weights of the original learning dynamics and the correction term. When $\lambda = 0$, StROL performs similar to Gradient as the corrective term has not effect on robot learning. In cases where the prior of known tasks is provided to the robot, the value of λ has not significant effect on the robot learning. However, when the robot is not provided with a prior of known tasks, StROL performs similar to the baelines when $0 < \lambda \leq 1$.

how would this change affect the robot learning?

We perform simulated experiments to evaluate the effect that the value of the hyperparameter λ has on the robot's performance. In this experiment, we evaluate the performane of the robot on four different values of λ: $0, 0.1, 1$, and 10. In addition to varying the value of this hyperparameter, we consider two learning scenarios: (1) *Best Case* scenario where the robot is provided with the prior of known tasks in the environment and tries to learn a the human trying to teach the robot has a similar amount of noise and bias in their actions as during training and (2) *Worst Case* scenario where the robot is provided with an incorrect prior of known tasks.

The results for this simulation are summarized in Figure A.1. We observe that when $\lambda = 0$, our proposed algorithm, StROL algorithm reduces to the Gradient approach (since the corrective term is multiplied by zero). In the best case scenario, we generally observe that

StROL is not particularly sensitive to changes in λ (when $\lambda > 0$). On the other hand, when the robot has incorrect information about the prior, in the worst case, we observe that as we increase the value of λ, the performance of StROL is affected and it slowly approaches the performance of e2e. Intuitively, this occurs because the corrective term dominated the original learning dynamics. We observe that the relative weight of g and $\hat{g}$ does not have a significant effect on the performance of StROL when the simulated human takes actions according to the prior of tasks. However, if the simulated human tries to teach a task different from priors, the baselines significantly outperform StROL with when $\lambda = 10$. However, when the relative weight of $\hat{g}$ is similar to that of g, StROL performs similar to the baselines.

A.2 Effect of Changing Human Preferences

In all of our experiments, we have assumed that the task that the user is trying to teach the robot remains the same throughout the duration of teaching. However, in reality, the user may change their task preferences midway while teaching the task. For example, the user may start teaching the robot to carry to the goal location while avoiding the laptop, but may later change their preference to moving over the laptop to save some time. How would the robot adapt to these changing human preferences?

To answer this question, we perform an additional simulation experiment where the user changes their preferences midway while teaching the task. In this simulation, the user provides a total of 5 inputs to the robot to teach their desired task. For the first 2 timesteps, they start teaching one task from the prior of known tasks to the robot. From the 3^{rd} timestep, they change their task preferences and start teaching a different task from the prior of known tasks.

The results for this simulation are summarized in Figure A.2. We observe that even when the

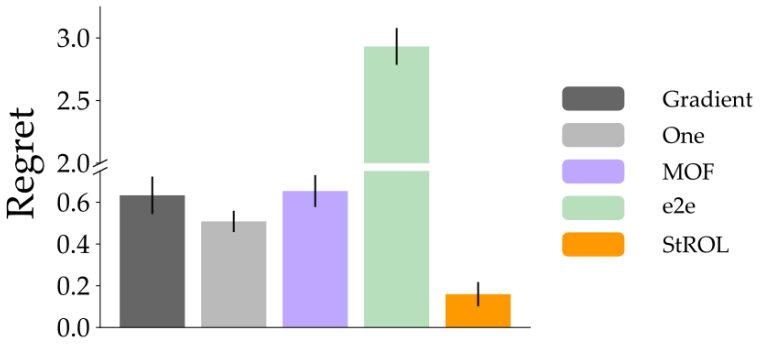


Figure A.2: Comparison of StROL with the baselines when the user changes preference for teaching a task midway through the interaction. The results show that StROL can adapt to the changing human preferences and learn the task correctly, leading to a lower regret.

simulated human changes the task preferences midway through the task, StROL can adapt to these changing preferences learn the task correctly. On the other hand, the baselines do not adapt to the changing preferences of the human in the limited number of interactions and lead to a higher regret as compared to StROL.

Appendix B

Additional Results for L2D2

B.1 User Study Subjective Responses

In addition to the subjective ratings provided by the users in the Likert scale survey, the users were given an option to fill out their subjective responses in paragraph format to highlight their experience in using these different teaching interfaces. These comments provided by the users for different teaching interfaces are provided in Table B.1.

The comments provided by the users show that they usually felt that teleoperating the robot (Teleop) was easy and intuitive when they had prior experience in using the controllers. However, the users without prior experience in using joysticks struggled with the interface. On the other hand, most users perceived drawing interfaces and easy to learn, and the process of demonstrating the tasks using drawing quicker than teleoperating the robot. When working with RT-Traj, the users indicated that while the drawing part made the teaching process easier, specifying the heights for the robot's workspace was unintuitive and led to incorrect task execution. In S2S, the users highlight that even though providing drawings is easier, providing a trajectory that conveys the same information in different camera frames was confusing. Finally, using L2D2, the users highlight that they had to spend less effort in terms of resetting the environment. The users also state that while the process of setting orientations was confusing at first, with practice they were able to effectively leverage the interface to demonstrate the tasks to the robot.

Table B.1: Open-Ended responses from participants in the user study that highlight their experience in using different demonstration and drawing interfaces to teach the robot.

Method	User Responses
Teleop	• *"Teleoperating the robot is simple and i know exactly what my actions were, but it requires time resetting and operating the robot"* • *"The interface was easy to use and I could quickly show the robot how to do the tasks. I did have to get up every time to change the environment which wasn't too bad but it did feel rather time consuming to repeatedly get up and change where the object was"* • *"Was not intuitive. I can not use the learning for a new task that easily"* • *"It was pretty intuitive, but easy to mess up. With practice could become second nature"*
RT-Traj	• *"The drawing interface is easy to use, but specifying the height of keypoints is not clear to me, there also seem to be an offset from the drawings to rollout which is hard to interpret"* • *"The robot was slow during rollouts and didn't seem to learn the task. It was also difficult to figure out exactly where the robot should go in the drawings to actually pick up or scoop the object on the table"* • *"It was very difficult. Could not get it to work"* • *"The interface was easy to understand but correlating the drawing with the numbers was a confusing process to visualize."*
S2S	• *"This interface is quick for providing drawings, although it requires me to physically move the objects, drawing on two cameras is confusing depending on where the object is placed"* • *"The interface was confusing at times because it was harder to imagine how the trajectory would look like from diffrerent angles. And for certain trajectories when drawing the trajectory for the second camera angle it was hard to show in 2D how the trajectory would look"* • *"It was very easy and user friendly. Easier to use than a joystick"* • *"Drawing the paths in the interface was made easier with the additional camera. It was easier to gauge the trajectory that the robot would take. I found the interface quite easy to use without much explanation"*
L2D2	• *"Providing drawings without having physically reset the environment everytime saves a lot of time, the interface is fine but the rotation process is a little unintuitive at first."* • *"It was quick and easy to provide demonstrations since I didn't have to get up to change the environment. The interface was intuitive to use and I didn't have to worry about imagining the trajectory from different angles or think too much about any factors other than the drawings"* • *"I enjoyed my experience using the interface. I found it easy to get comfortable with and found the task easy to complete after explanation."* • *"The surface touch screen was not good but the interface design was great. easy to learn and switch"*

Bibliography

[1] Dylan P Losey, Andrea Bajcsy, Marcia K O'Malley, and Anca D Dragan. Physical interaction as communication: Learning robot objectives online from human corrections. *The International Journal of Robotics Research*, 2021.

[2] Ashesh Jain, Shikhar Sharma, Thorsten Joachims, and Ashutosh Saxena. Learning preferences for manipulation tasks from online coactive feedback. *The International Journal of Robotics Research*, 34(10):1296–1313, 2015.

[3] Andreea Bobu, Marius Wiggert, Claire Tomlin, and Anca D Dragan. Inducing structure in reward learning by learning features. *The International Journal of Robotics Research*, 2022.

[4] Erdem Bıyık, Dylan P Losey, Malayandi Palan, Nicholas C Landolfi, Gleb Shevchuk, and Dorsa Sadigh. Learning reward functions from diverse sources of human feedback: Optimally integrating demonstrations and preferences. *The International Journal of Robotics Research*, 2021.

[5] Dylan P Losey, Andrea Bajcsy, Marcia K O'Malley, and Anca D Dragan. Physical interaction as communication: Learning robot objectives online from human corrections. *IJRR*, 41(1):20–44, 2022.

[6] Andreea Bobu, Andrea Bajcsy, Jaime F Fisac, Sampada Deglurkar, and Anca D Dragan. Quantifying hypothesis space misspecification in learning from human–robot demonstrations and physical corrections. *IEEE Transactions on Robotics*, 36(3):835–854, 2020.

[7] Peiqi Liu, Yaswanth Orru, Jay Vakil, Chris Paxton, Nur Muhammad Mahi Shafiullah, and Lerrel Pinto. Ok-robot: What really matters in integrating open-knowledge models for robotics. *arXiv preprint arXiv:2401.12202*, 2024.

[8] Yuta Sugiura, Daisuke Sakamoto, Anusha Withana, Masahiko Inami, and Takeo Igarashi. Cooking with robots: designing a household system working in open environments. In *Proceedings of the SIGCHI conference on human factors in computing systems*, pages 2427–2430, 2010.

[9] Ananth Jonnavittula, Shaunak A Mehta, and Dylan P Losey. Sari: Shared autonomy across repeated interaction. *ACM Transactions on Human-Robot Interaction*, 2024.

[10] Sven Nyholm and Jilles Smids. Automated cars meet human drivers: responsible human-robot coordination and the ethics of mixed traffic. *Ethics and Information Technology*, 22(4):335–344, 2020.

[11] Abdoulaye O Ly and Moulay Akhloufi. Learning to drive by imitation: An overview of deep behavior cloning methods. *IEEE Transactions on Intelligent Vehicles*, 6(2):195–209, 2020.

[12] Judy F Rosenblith. Learning by imitation in kindergarten children. *Child Development*, pages 69–80, 1959.

[13] Paul Bakker, Yasuo Kuniyoshi, et al. Robot see, robot do: An overview of robot imitation. In *AISB96 Workshop on Learning in Robots and Animals*, volume 5, pages 3–11, 1996.

[14] Stefan Schaal. Is imitation learning the route to humanoid robots? *Trends in cognitive sciences*, 3(6):233–242, 1999.

[15] Jonathan Spencer, Sanjiban Choudhury, Matthew Barnes, Matthew Schmittle, Mung Chiang, Peter Ramadge, and Sidd Srinivasa. Expert intervention learning. *Autonomous Robots*, 46(1):99–113, 2022.

[16] Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *International Conference on Artificial Intelligence and Statistics*, pages 627–635, 2011.

[17] Hongyi Liu and Lihui Wang. Gesture recognition for human-robot collaboration: A review. *International Journal of Industrial Ergonomics*, 68:355–367, 2018.

[18] Theodore R Sumers, Mark K Ho, Robert D Hawkins, Karthik Narasimhan, and Thomas L Griffiths. Learning rewards from linguistic feedback. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 6002–6010, 2021.

[19] Soheil Habibian, Antonio Alvarez Valdivia, Laura H Blumenschein, and Dylan P Losey. A review of communicating robot learning during human-robot interaction. *arXiv preprint arXiv:2312.00948*, 2023.

[20] Pim De Haan, Dinesh Jayaraman, and Sergey Levine. Causal confusion in imitation learning. In *Neural Information Processing Systems*, 2019.

[21] Jonathan Spencer, Sanjiban Choudhury, Arun Venkatraman, Brian Ziebart, and J Andrew Bagnell. Feedback in imitation learning: The three regimes of covariate shift. *arXiv preprint arXiv:2102.02872*, 2021.

[22] Liyiming Ke, Yunchu Zhang, Abhay Deshpande, Siddhartha Srinivasa, and Abhishek Gupta. CCIL: Continuity-based data augmentation for corrective imitation learning. In *International Conference on Learning Representations*, 2024.

[23] Allan Zhou, Moo Jin Kim, Lirui Wang, Pete Florence, and Chelsea Finn. NeRF in the palm of your hand: Corrective augmentation for robotics via novel-view synthesis. In *IEEE/CVF CVPR*, 2023.

[24] Zipeng Fu, Tony Z Zhao, and Chelsea Finn. Mobile ALOHA: Learning bimanual mobile manipulation using low-cost whole-body teleoperation. In *Conference on Robot Learning*, 2024.

[25] Philipp Wu, Yide Shentu, Zhongke Yi, Xingyu Lin, and Pieter Abbeel. GELLO: A general, low-cost, and intuitive teleoperation framework for robot manipulators. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 12156–12163, 2024.

[26] Aadhithya Iyer, Zhuoran Peng, Yinlong Dai, Irmak Guzey, Siddhant Haldar, Soumith Chintala, and Lerrel Pinto. OPEN TEACH: A versatile teleoperation system for robotic manipulation. In *Conference on Robot Learning*, 2024.

[27] Ananth Jonnavittula, Sagar Parekh, and Dylan P Losey. VIEW: Visual imitation learning with waypoints. *Autonomous Robots*, 49(1):1–26, 2025.

[28] Vidhi Jain, Maria Attarian, Nikhil J Joshi, Ayzaan Wahid, Danny Driess, Quan Vuong, Pannag R Sanketi, Pierre Sermanet, Stefan Welker, Christine Chan, et al. Vid2Robot: End-to-end video-conditioned policy learning with cross-attention transformers. In *Robotics: Science and Systems (RSS)*, 2024.

[29] Peihong Yu, Amisha Bhaskar, Anukriti Singh, Zahiruddin Mahammad, and Pratap Tokekar. Sketch-to-Skill: Bootstrapping robot learning with human drawn trajectory sketches. *arXiv preprint arXiv:2503.11918*, 2025.

[30] Jiayuan Gu, Sean Kirmani, Paul Wohlhart, Yao Lu, Montserrat Gonzalez Arenas,

Kanishka Rao, Wenhao Yu, Chuyuan Fu, Keerthana Gopalakrishnan, Zhuo Xu, Priya Sundaresan, Peng Xu, Hao Su, Karol Hausman, Chelsea Finn, Quan Vuong, and Ted Xiao. RT-trajectory: Robotic task generalization via hindsight trajectory sketches. In *International Conference on Learning Representations*, 2024.

[31] Shaunak A Mehta and Dylan P Losey. Unified learning from demonstrations, corrections, and preferences during physical human–robot interaction. *ACM Transactions on Human-Robot Interaction*, 13(3), 2024.

[32] Shaunak A Mehta, Forrest Meng, Andrea Bajcsy, and Dylan P Losey. StROL: Stabilized and robust online learning from humans. *IEEE Robotics and Automation Letters*, 2024.

[33] Shaunak A Mehta, Yusuf Umut Ciftci, Balamurugan Ramachandran, Somil Bansal, and Dylan P Losey. Stable-bc: Controlling covariate shift with stable behavior cloning. *IEEE Robotics and Automation Letters*, 2025.

[34] Shaunak A Mehta, Heramb Nemlekar, Hari Sumant, and Dylan P Losey. L2d2: Robot learning from 2d drawings. *arXiv preprint arXiv:2505.12072*, 2025.

[35] Neville Hogan. Impedance control: An approach to manipulation. In *American Control Conference*, pages 304–313, 1984.

[36] Agostino De Santis, Bruno Siciliano, Alessandro De Luca, and Antonio Bicchi. An atlas of physical human–robot interaction. *Mechanism and Machine Theory*, 43(3):253–270, 2008.

[37] Sami Haddadin and Elizabeth Croft. Physical human–robot interaction. In *Springer Handbook of Robotics*, pages 1835–1874. 2016.

[38] Dylan P Losey, Craig G McDonald, Edoardo Battaglia, and Marcia K O'Malley. A review of intent detection, arbitration, and communication aspects of shared control for physical human–robot interaction. *Applied Mechanics Reviews*, 70(1), 2018.

[39] Sami Haddadin, Alin Albu-Schaffer, Alessandro De Luca, and Gerd Hirzinger. Collision detection and reaction: A contribution to safe physical human-robot interaction. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3356–3363, 2008.

[40] Yanan Li, Gerolamo Carboni, Franck Gonzalez, Domenico Campolo, and Etienne Burdet. Differential game theory for versatile physical human–robot interaction. *Nature Machine Intelligence*, 1(1):36–43, 2019.

[41] Selma Musić and Sandra Hirche. Control sharing in human-robot team interaction. *Annual Reviews in Control*, 44:342–354, 2017.

[42] Alexander Mörtl, Martin Lawitzky, Ayse Kucukyilmaz, Metin Sezgin, Cagatay Basdogan, and Sandra Hirche. The role of roles: Physical cooperation between humans and robots. *The International Journal of Robotics Research*, 31(13):1656–1674, 2012.

[43] Brenna D Argall, Sonia Chernova, Manuela Veloso, and Brett Browning. A survey of robot learning from demonstration. *Robotics and Autonomous Systems*, 57(5):469–483, 2009.

[44] Baris Akgun, Maya Cakmak, Karl Jiang, and Andrea L Thomaz. Keyframe-based learning from demonstration. *International Journal of Social Robotics*, 4(4):343–355, 2012.

[45] Dylan P Losey and Marcia K O'Malley. Learning the correct robot trajectory in

real-time from physical human interactions. *ACM Transactions on Human-Robot Interaction*, 9(1):1–19, 2019.

[46] Mahdi Khoramshahi and Aude Billard. A dynamical system approach to task-adaptation in physical human–robot interaction. *Autonomous Robots*, 43(4):927–946, 2019.

[47] Michael Hagenow, Emmanuel Senft, Robert Radwin, Michael Gleicher, Bilge Mutlu, and Michael Zinn. Corrective shared autonomy for addressing task variability. *IEEE Robotics and Automation Letters*, 6(2):3720–3727, 2021.

[48] Takayuki Osa, Joni Pajarinen, Gerhard Neumann, J Andrew Bagnell, Pieter Abbeel, Jan Peters, et al. An algorithmic perspective on imitation learning. *Foundations and Trends in Robotics*, 7(1-2):1–179, 2018.

[49] Pieter Abbeel and Andrew Y Ng. Apprenticeship learning via inverse reinforcement learning. In *International Conference on Machine Learning*, 2004.

[50] Andrew Y Ng and Stuart J Russell. Algorithms for inverse reinforcement learning. In *International Conference on Machine Learning*, 2000.

[51] Mrinal Kalakrishnan, Peter Pastor, Ludovic Righetti, and Stefan Schaal. Learning objective functions for manipulation. In *IEEE International Conference on Robotics and Automation*, pages 1331–1336, 2013.

[52] Hang Yin, Francisco S Melo, Ana Paiva, and Aude Billard. An ensemble inverse optimal control approach for robotic task learning and adaptation. *Autonomous Robots*, 43(4):875–896, 2019.

[53] Mengxi Li, Alper Canberk, Dylan P Losey, and Dorsa Sadigh. Learning human ob-

jectives from sequences of physical corrections. In *IEEE International Conference on Robotics and Automation*, pages 2877–2883, 2021.

[54] Michael Kelly, Chelsea Sidrane, Katherine Driggs-Campbell, and Mykel J Kochenderfer. HG-DAgger: Interactive imitation learning with human experts. In *International Conference on Robotics and Automation*, pages 8077–8083, 2019.

[55] Ryan Hoque, Ashwin Balakrishna, Ellen Novoseller, Albert Wilcox, Daniel S Brown, and Ken Goldberg. ThriftyDAgger: Budget-aware novelty and risk gating for interactive imitation learning. In *Conference on Robot Learning*, 2021.

[56] Borja Ibarz, Jan Leike, Tobias Pohlen, Geoffrey Irving, Shane Legg, and Dario Amodei. Reward learning from human preferences and demonstrations in Atari. In *Advances in Neural Information Processing Systems*, 2018.

[57] Daniel Brown, Wonjoon Goo, Prabhat Nagarajan, and Scott Niekum. Extrapolating beyond suboptimal demonstrations via inverse reinforcement learning from observations. In *International Conference on Machine Learning*, pages 783–792, 2019.

[58] Letian Chen, Rohan Paleja, and Matthew Gombolay. Learning from suboptimal demonstration via self-supervised reward regression. In *Conference on Robot Learning*, pages 1262–1277, 2021.

[59] Songyuan Zhang, Zhangjie Cao, Dorsa Sadigh, and Yanan Sui. Confidence-aware imitation learning from demonstrations with varying optimality. In *Advances in Neural Information Processing Systems*, 2021.

[60] Kimin Lee, Laura M Smith, and Pieter Abbeel. PEBBLE: Feedback-efficient interactive reinforcement learning via relabeling experience and unsupervised pre-training. In *International Conference on Machine Learning*, pages 6152–6163, 2021.

[61] Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences. In *Advances in Neural Information Processing Systems*, 2017.

[62] Hong Jun Jeon, Smitha Milli, and Anca Dragan. Reward-rational (implicit) choice: A unifying formalism for reward learning. In *Advances in Neural Information Processing Systems*, pages 4415–4426, 2020.

[63] Shaunak A Mehta, Soheil Habibian, and Dylan P Losey. Waypoint-based reinforcement learning for robot manipulation tasks. In *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 541–548. IEEE, 2024.

[64] Maegan Tucker, Ellen Novoseller, Claudia Kann, Yanan Sui, Yisong Yue, Joel W Burdick, and Aaron D Ames. Preference-based learning for exoskeleton gait optimization. In *ICRA*, pages 2351–2357, 2020.

[65] Klas Kronander and Aude Billard. Online learning of varying stiffness through physical human-robot interaction. In *ICRA*, 2012.

[66] Wanxin Jin, Todd D Murphey, Zehui Lu, and Shaoshuai Mou. Learning from human directional corrections. *IEEE Transactions on Robotics*, 2022.

[67] Nathan D Ratliff, J Andrew Bagnell, and Martin A Zinkevich. Maximum margin planning. In *ICML*, pages 729–736, 2006.

[68] Alexander Broad, Ian Abraham, Todd Murphey, and Brenna Argall. Data-driven koopman operators for model-based shared control of human–machine systems. *IJRR*, 39(9):1178–1195, 2020.

[69] Ran Tian, Masayoshi Tomizuka, Anca D Dragan, and Andrea Bajcsy. Towards mod-

eling and influencing the dynamics of human learning. In *ACM/IEEE International Conference on Human-Robot Interaction*, 2023.

[70] Matteo Saveriano, Fares J Abu-Dakka, Aljaz Kramberger, and Luka Peternel. Dynamic movement primitives in robotics: A tutorial survey. *arXiv preprint arXiv:2102.03861*, 2021.

[71] Mark A Thornton and Diana I Tamir. People accurately predict the transition probabilities between actions. *Science Advances*, 2021.

[72] Rohin Shah, Dmitrii Krasheninnikov, Jordan Alexander, Pieter Abbeel, and Anca Dragan. Preferences implicit in the state of the world. *ICLR*, 2019.

[73] Rachit Dubey, Pulkit Agrawal, Deepak Pathak, Thomas L Griffiths, and Alexei A Efros. Investigating human priors for playing video games. *arXiv preprint arXiv:1802.10217*, 2018.

[74] Chris L Baker and Joshua B Tenenbaum. Modeling human plan recognition using bayesian theory of mind. *Plan, Activity, and Intent Recognition: Theory and practice*, 7:177–204, 2014.

[75] Bowen Zhang and Harold Soh. Large language models as zero-shot human models for human-robot interaction. *arXiv preprint arXiv:2303.03548*, 2023.

[76] Dean A Pomerleau. Alvinn: An autonomous land vehicle in a neural network. In *Neural Information Processing Systems*, 1988.

[77] Rafael Figueiredo Prudencio, Marcos ROA Maximo, and Esther Luna Colombini. A survey on offline reinforcement learning: Taxonomy, review, and open problems. *IEEE Transactions on Neural Networks and Learning Systems*, 2023.

[78] Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative q-learning for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 33:1179–1191, 2020.

[79] Michael Laskey, Jonathan Lee, Roy Fox, Anca Dragan, and Ken Goldberg. DART: Noise injection for robust imitation learning. In *Conference on Robot Learning*, 2017.

[80] Yusuf Umut Ciftci, Zeyuan Feng, and Somil Bansal. SAFE-GIL: Safety guided imitation learning. *arXiv preprint arXiv:2404.05249*, 2024.

[81] Liyiming Ke, Jingqiang Wang, Tapomayukh Bhattacharjee, Byron Boots, and Siddhartha Srinivasa. Grasping with chopsticks: Combating covariate shift in model-free imitation learning for fine manipulation. In *IEEE International Conference on Robotics and Automation*, 2021.

[82] Jonathan D Chang, Masatoshi Uehara, Dhruv Sreenivas, Rahul Kidambi, and Wen Sun. Mitigating covariate shift in imitation learning via offline data without great coverage. In *Neural Information Processing Systems*, 2021.

[83] Ryan Hoque, Ajay Mandlekar, Caelan Garrett, Ken Goldberg, and Dieter Fox. IntervenGen: Interventional data generation for robust and data-efficient robot imitation learning. *arXiv:2405.01472*, 2024.

[84] Kunal Menda, Katherine Driggs-Campbell, and Mykel J Kochenderfer. EnsembleDAgger: A bayesian approach to safe imitation learning. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2019.

[85] Sheng Yue, Xingyuan Hua, Ju Ren, Sen Lin, Junshan Zhang, and Yaoxue Zhang. OLLIE: Imitation learning from offline pretraining to online finetuning. *arXiv preprint arXiv:2405.17477*, 2024.

[86] Allen Ren, Sushant Veer, and Anirudha Majumdar. Generalization guarantees for imitation learning. In *Conference on Robot Learning*, 2021.

[87] Kim P Wabersich, Andrew J Taylor, Jason J Choi, Koushil Sreenath, Claire J Tomlin, Aaron D Ames, and Melanie N Zeilinger. Data-driven safety filters: Hamilton-Jacobi reachability, control barrier functions, and predictive methods for uncertain systems. *IEEE Control Systems Mag.*, 2023.

[88] Kai-Chieh Hsu, Haimin Hu, and Jaime F Fisac. The safety filter: A unified view of safety-critical control in autonomous systems. *Annual Review of Control, Robotics, and Autonomous Systems*, 2023.

[89] Alfredo Reichlin, Giovanni Luca Marchetti, Hang Yin, Ali Ghadirzadeh, and Danica Kragic. Back to the manifold: Recovering from out-of-distribution states. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2022.

[90] Dionis Totsila, Konstantinos Chatzilygeroudis, Denis Hadjivelichkov, Valerio Modugno, Ioannis Hatzilygeroudis, and Dimitrios Kanoulas. End-to-end stable imitation learning via autonomous neural dynamic policies. In *IEEE International Conference on Robotics and Automation*, 2023.

[91] S Mohammad Khansari-Zadeh and Aude Billard. Learning stable nonlinear dynamical systems with gaussian mixture models. *IEEE Transactions on Robotics*, 2011.

[92] Daniel Pfrommer, Thomas Zhang, Stephen Tu, and Nikolai Matni. TaSIL: Taylor series imitation learning. In *Neural Information Processing Systems*, 2022.

[93] Katie Kang, Paula Gradu, Jason J Choi, Michael Janner, Claire Tomlin, and Sergey Levine. Lyapunov density models: Constraining distribution shift in learning-based control. In *Int. Conf. on Machine Learning*, 2022.

[94] Suneel Belkhale, Yuchen Cui, and Dorsa Sadigh. Data quality in imitation learning. *Advances in Neural Information Processing Systems*, pages 80375–80395, 2023.

[95] Harish Ravichandar, Athanasios S Polydoros, Sonia Chernova, and Aude Billard. Recent advances in robot learning from demonstration. *Annual Review of Control, Robotics, and Autonomous Systems*, 3(1):297–330, 2020.

[96] Hongjie Fang, Hao-Shu Fang, Yiming Wang, Jieji Ren, Jingjing Chen, Ruo Zhang, Weiming Wang, and Cewu Lu. AirExo: Low-cost exoskeletons for learning whole-arm manipulation in the wild. In *IEEE International Conference on Robotics and Automation*, pages 15031–15038, 2024.

[97] Xiaoyu Zhang, Matthew Chang, Pranav Kumar, and Saurabh Gupta. Diffusion meets DAgger: Supercharging eye-in-hand imitation learning. In *Robotics: Science and Systems*, 2024.

[98] Eric Jang, Alex Irpan, Mohi Khansari, Daniel Kappler, Frederik Ebert, Corey Lynch, Sergey Levine, and Chelsea Finn. BC-Z: Zero-shot task generalization with robotic imitation learning. In *Conference on Robot Learning*, pages 991–1002, 2022.

[99] Ling-Huan Kong, Wei He, Wen-Shi Chen, Hui Zhang, and Yao-Nan Wang. Dynamic movement primitives based robot skills learning. *Machine Intelligence Research*, 20(3):396–407, 2023.

[100] Jihong Zhu, Michael Gienger, and Jens Kober. Learning task-parameterized skills from few demonstrations. *IEEE Robotics and Automation Letters*, 7(2):4063–4070, 2022.

[101] Tian Gao, Soroush Nasiriany, Huihan Liu, Quantao Yang, and Yuke Zhu. Prime: Scaffolding manipulation tasks with behavior primitives for data-efficient imitation learning. *IEEE Robotics and Automation Letters*, 2024.

[102] Neo Ee Sian, Kazuhito Yokoi, Shuuji Kajita, Fumio Kanehiro, and Kazuo Tanie. Whole body teleoperation of a humanoid robot development of a simple master device using joysticks. *Journal of the Robotics Society of Japan*, 22(4):519–527, 2004.

[103] Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. LIBERO: Benchmarking knowledge transfer for lifelong robot learning. *Advances in Neural Information Processing Systems*, 2024.

[104] Shuran Song, Andy Zeng, Johnny Lee, and Thomas Funkhouser. Grasping in the wild: Learning 6DOF closed-loop grasping from low-cost demonstrations. *IEEE Robotics and Automation Letters*, 5(3):4978–4985, 2020.

[105] Shaunak A Mehta, Sagar Parekh, and Dylan P Losey. Learning latent actions without human demonstrations. In *IEEE International Conference on Robotics and Automation*, pages 7437–7443, 2022.

[106] Tianhao Zhang, Zoe McCarthy, Owen Jow, Dennis Lee, Xi Chen, Ken Goldberg, and Pieter Abbeel. Deep imitation learning for complex manipulation tasks from virtual reality teleoperation. In *IEEE International Conference on Robotics and Automation*, pages 5628–5635, 2018.

[107] Abraham George, Alison Bartsch, and Amir Barati Farimani. Openvr: Teleoperation for manipulation. *SoftwareX*, 29:102054, 2025.

[108] Jiafei Duan, Yi Ru Wang, Mohit Shridhar, Dieter Fox, and Ranjay Krishna. Ar2-d2: Training a robot without a robot. In *7th Annual Conference on Robot Learning*.

[109] Reihaneh Mirjalili, Tobias Jülg, Florian Walter, and Wolfram Burgard. Augmented reality for robots (arro): Pointing visuomotor policies towards visual robustness. *arXiv preprint arXiv:2505.08627*, 2025.

[110] Minttu Alakuijala, Gabriel Dulac-Arnold, Julien Mairal, Jean Ponce, and Cordelia Schmid. Learning reward functions for robotic manipulation by observing humans. In *IEEE International Conference on Robotics and Automation*, pages 5006–5012, 2023.

[111] Shikhar Bahl, Abhinav Gupta, and Deepak Pathak. Human-to-robot imitation in the wild. In *Robotics: Science and Systems*, 2022.

[112] Sarah Young, Dhiraj Gandhi, Shubham Tulsiani, Abhinav Gupta, Pieter Abbeel, and Lerrel Pinto. Visual imitation made easy. In *Conference on Robot Learning*, pages 1992–2005, 2021.

[113] Nur Muhammad Mahi Shafiullah, Anant Rai, Haritheja Etukuru, Yiqian Liu, Ishan Misra, Soumith Chintala, and Lerrel Pinto. On bringing robots home. *arXiv preprint arXiv:2311.16098*, 2023.

[114] Homanga Bharadhwaj, Roozbeh Mottaghi, Abhinav Gupta, and Shubham Tulsiani. Track2Act: Predicting point tracks from internet videos enables generalizable robot manipulation. In *European Conference on Computer Vision*, pages 306–324, 2024.

[115] Frederik Ebert, Yanlai Yang, Karl Schmeckpeper, Bernadette Bucher, Georgios Georgakis, Kostas Daniilidis, Chelsea Finn, and Sergey Levine. Bridge data: Boosting generalization of robotic skills with cross-domain datasets. In *Robotics: Science and Systems (RSS)*, 2022.

[116] Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Jasmine Hsu, et al. RT-1: Robotics transformer for real-world control at scale. *arXiv preprint arXiv:2212.06817*, 2022.

[117] Abby O'Neill, Abdul Rehman, Abhiram Maddukuri, Abhishek Gupta, Abhishek

Padalkar, Abraham Lee, Acorn Pooley, Agrim Gupta, Ajay Mandlekar, Ajinkya Jain, et al. Open X-Embodiment: Robotic learning datasets and RT-X models. In *IEEE International Conference on Robotics and Automation*, pages 6892–6903, 2024.

[118] Sudeep Dasari and Abhinav Gupta. Transformers for one-shot visual imitation. In *Conference on Robot Learning*, pages 2071–2084, 2021.

[119] Stephen James, Michael Bloesch, and Andrew J Davison. Task-embedded control networks for few-shot imitation learning. In *Conference on Robot Learning*, pages 783–795, 2018.

[120] Yan Duan, Marcin Andrychowicz, Bradly Stadie, OpenAI Jonathan Ho, Jonas Schneider, Ilya Sutskever, Pieter Abbeel, and Wojciech Zaremba. One-shot imitation learning. *Advances in Neural Information Processing Systems (NeurIPS)*, 30, 2017.

[121] Corey Lynch and Pierre Sermanet. Language conditioned imitation learning over unstructured data. In *Robotics: Science and Systems*, 2021.

[122] Khanh Nguyen, Debadeepta Dey, Chris Brockett, and Bill Dolan. Vision-based navigation with language-based assistance via imitation learning with indirect intervention. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12527–12537, 2019.

[123] Simon Stepputtis, Joseph Campbell, Mariano Phielipp, Stefan Lee, Chitta Baral, and Heni Ben Amor. Language-conditioned imitation learning for robot manipulation tasks. *Advances in Neural Information Processing Systems*, pages 13139–13150, 2020.

[124] Ryan Hoque, Ajay Mandlekar, Caelan Reed Garrett, Ken Goldberg, and Dieter Fox. Interventional data generation for robust and data-efficient robot imitation learning.

In *First Workshop on Out-of-Distribution Generalization in Robotics at CoRL 2023*, 2023.

[125] Jyothish Pari, Nur Muhammad Mahi Shafiullah, Sridhar Pandian Arunachalam, and Lerrel Pinto. The surprising effectiveness of representation learning for visual imitation. In *Robotics: Science and Systems*, 2022.

[126] Zhao Mandi, Homanga Bharadhwaj, Vincent Moens, Shuran Song, Aravind Rajeswaran, and Vikash Kumar. CACTI: A framework for scalable multi-task multi-scene visual imitation learning. In *CoRL 2022 Workshop on Pre-Training Robot Learning*, 2022.

[127] Tianhe Yu, Ted Xiao, Austin Stone, Jonathan Tompson, Anthony Brohan, Su Wang, Jaspiar Singh, Clayton Tan, Jodilyn Peralta, Brian Ichter, et al. Scaling robot learning with semantically imagined experience. 2023.

[128] Priya Sundaresan, Quan Vuong, Jiayuan Gu, Peng Xu, Ted Xiao, Sean Kirmani, Tianhe Yu, Michael Stark, Ajinkya Jain, Karol Hausman, Dorsa Sadigh, Jeannette Bohg, and Stefan Schaal. RT-sketch: Goal-conditioned imitation learning from hand-drawn sketches. In *Conference on Robot Learning*, 2024.

[129] Kosei Tanada, Yuka Iwanaga, Masayoshi Tsuchinaga, Yuji Nakamura, Takemitsu Mori, Remi Sakai, and Takashi Yamamoto. Sketch-MoMa: Teleoperation for mobile manipulator via interpretation of hand-drawn sketches. *arXiv preprint arXiv:2412.19153*, 2024.

[130] Weiming Zhi, Tianyi Zhang, and Matthew Johnson-Roberson. Instructing robots by sketching: Learning from demonstration via probabilistic diagrammatic teaching. In *IEEE International Conference on Robotics and Automation*, pages 15047–15053, 2024.

[131] Danelle Shah, Joseph Schneider, and Mark Campbell. A sketch interface for robust and natural robot control. *Proceedings of the IEEE*, 100(3):604–622, 2012.

[132] Brian D Ziebart, Andrew L Maas, J Andrew Bagnell, and Anind K Dey. Maximum entropy inverse reinforcement learning. In *AAAI*, pages 1433–1438, 2008.

[133] Anca D Dragan, Katharina Muelling, J Andrew Bagnell, and Siddhartha S Srinivasa. Movement primitives via optimization. In *IEEE International Conference on Robotics and Automation*, pages 2339–2346, 2015.

[134] Dylan P Losey and Marcia K O'Malley. Trajectory deformations from physical human–robot interaction. *IEEE Transactions on Robotics*, 34(1):126–138, 2017.

[135] R Duncan Luce. *Individual Choice Behavior: A Theoretical Analysis*. Courier Corporation, 2012.

[136] Christopher Lucas, Thomas Griffiths, Fei Xu, and Christine Fawcett. A rational model of preference learning and choice prediction by children. In *Advances in Neural Information Processing Systems*, 2008.

[137] Roger N Shepard. Stimulus and response generalization: A stochastic model relating generalization to distance in psychological space. *Psychometrika*, 22(4):325–345, 1957.

[138] Thomas M Cover. *Elements of Information Theory*. John Wiley & Sons, 1999.

[139] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[140] Taylor A Howell, Brian E Jackson, and Zachary Manchester. ALTRO: A fast solver for constrained trajectory optimization. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 7674–7679, 2019.

[141] John Schulman, Yan Duan, Jonathan Ho, Alex Lee, Ibrahim Awwal, Henry Bradlow, Jia Pan, Sachin Patil, Ken Goldberg, and Pieter Abbeel. Motion planning with sequential convex optimization and convex collision checking. *The International Journal of Robotics Research*, 33(9):1251–1270, 2014.

[142] Philip E Gill, Walter Murray, and Michael A Saunders. SNOPT: An SQP algorithm for large-scale constrained optimization. *SIAM review*, 47(1):99–131, 2005.

[143] Justin Fu, Katie Luo, and Sergey Levine. Learning robust rewards with adversarial inverse reinforcement learning. In *International Conference on Learning Representations*, 2018.

[144] Mariah L Schrum, Michael Johnson, Muyleng Ghuy, and Matthew C Gombolay. Four years in review: Statistical practices of likert scales in human-robot interaction studies. In *Companion of the 2020 ACM/IEEE International Conference on Human-Robot Interaction*, pages 43–52, 2020.

[145] Daniel S Brown, Wonjoon Goo, and Scott Niekum. Better-than-demonstrator imitation learning via automatically-ranked demonstrations. In *Conference on robot learning*, pages 330–359. PMLR, 2020.

[146] Mariah L Schrum, Erin Hedlund-Botti, Nina Moorman, and Matthew C Gombolay. Mind meld: Personalized meta-learning for robot-centric imitation learning. In *2022 17th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pages 157–165. IEEE, 2022.

[147] Avi Singh, Huihan Liu, Gaoyue Zhou, Albert Yu, Nicholas Rhinehart, and Sergey Levine. Parrot: Data-driven behavioral priors for reinforcement learning. *arXiv preprint arXiv:2011.10024*, 2020.

[148] Andrey Rudenko, Luigi Palmieri, Johannes Doellinger, Achim J Lilienthal, and Kai O Arras. Learning occupancy priors of human motion from semantic maps of urban environments. *IEEE RA-L*, 2021.

[149] Andrea Bajcsy, Anand Siththaranjan, Claire J Tomlin, and Anca D Dragan. Analyzing human models that adapt online. In *ICRA*, 2021.

[150] Ariel Rubinstein. *Modeling bounded rationality*. MIT press, 1998.

[151] Thomas L Griffiths, Falk Lieder, and Noah D Goodman. Rational use of cognitive resources: Levels of analysis between the computational and the algorithmic. *Topics in Cognitive Science*, 2015.

[152] Hassan K. Khalil. *Nonlinear Systems*, volume 3. Prentice Hall, 2008.

[153] Zhangjie Cao, Erdem Biyik, Woodrow Z. Wang, Allan Raventos, Adrien Gaidon, Guy Rosman, and Dorsa Sadigh. Reinforcement learning based control of imitative policies for near-accident driving. In *RSS*, July 2020.

[154] Siddarth Jain and Brenna Argall. Probabilistic human intent recognition for shared autonomy in assistive robotics. *ACM Transactions on Human-Robot Interaction (THRI)*, 9(1):1–23, 2019.

[155] Stéphane Ross and Drew Bagnell. Efficient reductions for imitation learning. In *Int. Conf. on Artificial Intelligence and Statistics*, 2010.

[156] Gerald Teschl. *Ordinary Differential Equations and Dynamical Systems*. American Mathematical Society, 2012.

[157] Xingyi Zhou, Rohit Girdhar, Armand Joulin, Philipp Krähenbühl, and Ishan Misra. Detecting twenty-thousand classes using image-level supervision. In *European Conference on Computer Vision*, pages 350–368, 2022.

[158] Xu Zhao, Wenchao Ding, Yongqi An, Yinglong Du, Tao Yu, Min Li, Ming Tang, and Jinqiao Wang. Fast segment anything. *arXiv preprint arXiv:2306.12156*, 2023.

[159] Richard Hartley and Andrew Zisserman. *Multiple view geometry in computer vision*. Cambridge University Press, 2003.

[160] Ian T Jolliffe and Jorge Cadima. Principal component analysis: A review and recent developments. *Philosophical transactions of the royal society A: Mathematical, Physical and Engineering Sciences*.

[161] Nitish Bahadur, Brian Lewandowski, and Randy Paffenroth. Dimension estimation using autoencoders and application. *Deep Learning Applications*, pages 95–121, 2022.

[162] Michal Rolinek, Dominik Zietlow, and Georg Martius. Variational autoencoders pursue PCA directions (by accident). In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12406–12415, 2019.

[163] Quentin Fournier and Daniel Aloise. Empirical comparison between autoencoders and traditional dimensionality reduction methods. In *IEEE International Conference on Artificial Intelligence and Knowledge Engineering*, pages 211–214, 2019.

[164] Davide Cacciarelli and Murat Kulahci. Hidden dimensions of the data: PCA vs autoencoders. *Quality Engineering*, pages 741–750, 2023.

[165] Youyi Zheng, Xiang Chen, Ming-Ming Cheng, Kun Zhou, Shi-Min Hu, and Niloy J Mitra. Interactive images: Cuboid proxies for smart image manipulation. *ACM Transactions on Graphics*, 2012.

[166] Homanga Bharadhwaj, Debidatta Dwibedi, Abhinav Gupta, Shubham Tulsiani, Carl Doersch, Ted Xiao, Dhruv Shah, Fei Xia, Dorsa Sadigh, and Sean Kirmani. Gen2Act:

Human video generation in novel scenarios enables generalizable robot manipulation. In *1st Workshop on X-Embodiment Robot Learning*, 2024.

[167] Sagar Parekh, Heramb Nemlekar, and Dylan P Losey. Towards balanced behavior cloning from imbalanced datasets. *arXiv preprint arXiv:2508.06319*, 2025.

[168] Yifeng Zhu, Abhishek Joshi, Peter Stone, and Yuke Zhu. Viola: Imitation learning for vision-based manipulation with object proposal priors. In *Conference on Robot Learning*, pages 1199–1210. PMLR, 2023.

[169] Ajay Mandlekar, Danfei Xu, Roberto Martín-Martín, Silvio Savarese, and Li Fei-Fei. Learning to generalize across long-horizon tasks from human demonstrations. *arXiv preprint arXiv:2003.06085*, 2020.

[170] Chen Wang, Linxi Fan, Jiankai Sun, Ruohan Zhang, Li Fei-Fei, Danfei Xu, Yuke Zhu, and Anima Anandkumar. Mimicplay: Long-horizon imitation learning by watching human play. In *7th Annual Conference on Robot Learning*.