Aligning Robot Representations with Humans

By

Andreea Bobu

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Engineering – Electrical Engineering and Computer Sciences

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Associate Professor Anca Dragan, Chair
Professor Claire Tomlin
Professor Ken Goldberg
Associate Professor Maya Cakmak

Summer 2023

Aligning Robot Representations with Humans

Abstract

Aligning Robot Representations with Humans

by

Andreea Bobu

Doctor of Philosophy in Engineering – Electrical Engineering and Computer Sciences

University of California, Berkeley

Associate Professor Anca Dragan, Chair

Robots are becoming increasingly weaved into the fabric of our society, from self-driving cars on our streets to assistive manipulators in our homes. To act in the world, robots rely on a *representation* of salient features of the task: for example, to hand me a cup of coffee, the robot considers movement efficiency and cup orientation in its behavior. However, if we want robots to act *for and with people*, their representations must not be just functional but also reflective of what humans care about, i.e. their representations must be *aligned* with humans'. What's holding us back from successful human-robot interaction is that these representations are often *misaligned*, resulting in anything from miscoordination and misunderstandings, to learning and executing dangerous behaviors.

To learn the human's representation of what matters in a task, typical methods rely on data sets of human behavior but this data cannot reflect every individual, environment, and task the robot will be exposed to. This dissertation advocates that we should instead treat humans as active participants in the interaction not as static data sources: **robots must engage with humans in an interactive process for finding a shared representation**. We formalize the *representation alignment problem* as a joint search for a common representation. Then, rather than hoping that representations will naturally be aligned, we propose having humans directly teach them to robots with *representation-specific input*. Next, we enable robots to automatically *detect representation misalignment* with the human by estimating a confidence over how much the robot's representation can explain the human's behavior. We demonstrate how human-aligned representations can lead to *novel human behavior models* with broad implications beyond robotics, to econometrics and cognitive science. Finally, this thesis concludes by asking *"How can robots help the human-robot team converge to a shared representation?"* and discusses opportunities for future work in expanding representation alignment for seamless human-robot interaction.

To my parents, Liviu and Mărioara.

Contents

List of Figures

List of Tables

Acknowledgments

A long time ago, my father told me that we can't get anywhere in life without people. As I reflect back on my six years of PhD, that little piece of wisdom resonates more profoundly than ever – both in life *and* in research. Just like I concluded that robots need people to fulfill their potential, so too I realize I would not be standing at the finish line without those who have inspired, mentored, and supported me.

First, I want to thank my PhD advisor Anca Dragan. I could not have asked for a better mentor, and I am immensely grateful that she took me under her wing. Anca has been one of my biggest advocates, a constant source of inspiration, and an exemplary role model. Her mentorship has been instrumental for shaping me into the researcher that I am today, showing me how to navigate the uncertainties of research and science, and teaching me how to hone in on that "key insight", that "a-ha moment" that infuses life into a paper. Thank you, Anca, for everything that you've done to nurture my academic journey.

I am also incredibly thankful to my thesis committee members and the many faculty that have supported me professionally: thank you to Claire Tomlin, Dorsa Sadigh, Julie Shah, Ken Goldberg, Maya Cakmak, and Sonia Chernova for your invaluable guidance.

Lab culture plays a big role in one's PhD journey, so I would also like to thank the InterACT lab members for nurturing my growth and happiness over the years. In particular, thank you to Andrea Bajcsy for mentoring me and commiserating with me about the occasionally nebulous processes of academia, to Jaime Fisac for his infectious cheer, optimism and enthusiasm that always reassured and motivated me in research, to Rohin Shah for our collaborative brainstorming sessions during the depths of the pandemic, and to Micah Carroll for believing that I can be a good mentor. I also want to thank Daniel Brown for stepping up when the lab needed support the most.

I am incredibly grateful for my collaborations outside of lab and even beyond Berkeley. Thank you to Marius Wiggert for working literally around the clock with me during the pandemic – you in Germany and me in Berkeley. Thank you to Andi Peng for working together to crystallize vague abstract ideas into math and to Inês Lourenço for introducing me to the charm of deriving algorithms in Swedish coffee shops during the dead of winter. I am also deeply appreciative of Chris Paxton from my NVIDIA internship for his unwavering support. Lastly, I want to thank the junior students – Arjun Sripathy, Matthew Zurek, Sampada Deglurkar, Yi Liu, Regina Wang and David Zhang – for trusting me to mentor them in research. It has been a privilege to work with each of you.

Thank you to my amazing friends for always believing in me, through both the good and the bad: Vickie Ye, Greg Lubin, Andi Peng, Avi Singh, Otilia Stretcu, Suma Anand, Vitchyr Pong, Allan Jabri, Micah Carroll, Li Dayan, May Simpson, Chelsea Edwards, Paula Gradu, Allison Sharrar, Callie Davidson, Sebastian Garza, and Alexandra Barsan.

Last but not least, thank you to my parents, who have supported me my entire life, with anything I have wanted to do, even when it involved moving across the ocean. I would not be where I am today without your unwavering support, kindness, and warmth.

Chapter 1

Introduction

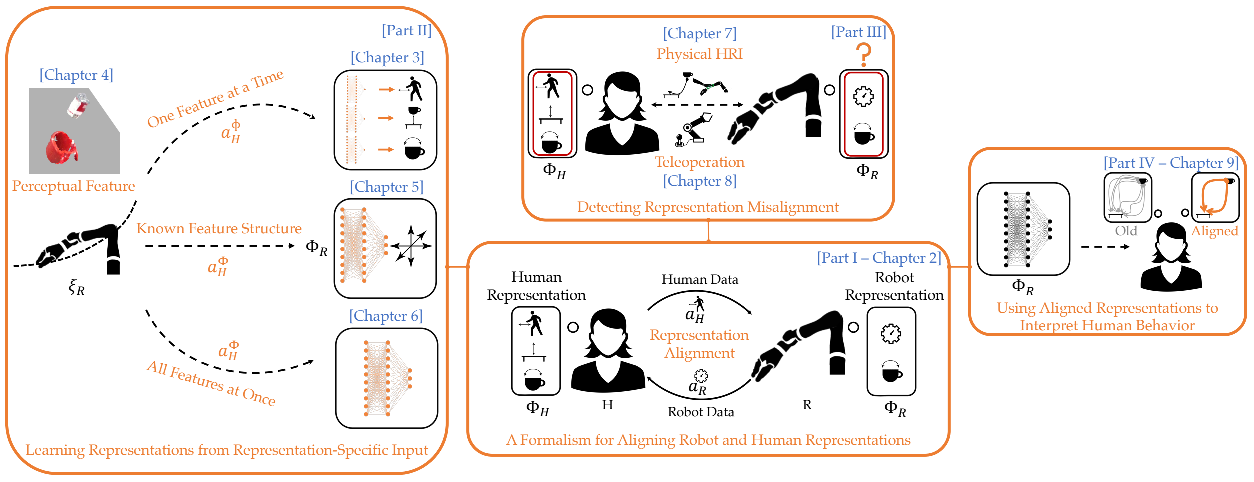


Figure 1.1: Thesis overview contextualizing the dissertation chapters in the broader space of representation alignment in human-robot interaction.

Robots are increasingly weaved into the fabric of society, from self-driving cars in cities to personal robots on space stations to assistive arms at home. Despite this rise of autonomous system integration, seamless robot interaction with people remains elusive: autonomous cars still cause crashes in unconventional situations [202], flight control systems create tug of war scenarios in spite of conflicting pilot interventions [105], and, as we will see in this thesis, personal robots move objects uncomfortably close to their users [43]. What is the reason underlying these failure modes in human-robot interaction? What's still missing in the way our robots currently decide how to behave around people?

To act in the world, robots rely on a *representation* of salient features of the task: for instance, to hand a user a cup of coffee, the robot may consider movement efficiency and cup orientation in its behavior. In this simple example, it's relatively easy to anticipate what the robot's representation should be: almost every system designer will be able to think ahead of time of efficiency and joint orientations as features important for general robot motion. However, human preferences for how robots should behave around them can be vastly more complex, varying between individuals, environments, and tasks. In the above example, a user might care about how close to them the robot moves or whether it carries the mugful of coffee around their expensive laptop, and what "close" or "around" means will differ from person to person and be context-dependent. This makes enumerating all the features that could matter to people much more challenging, even impossible.

This example illustrates that if we want robots to act *for and with people*, their representations must not be just functional but also reflective of what humans care about, i.e. their representations must be *aligned* with humans'. Once we adopt this perspective, we see that what's holding us back from successful human-robot interaction is that these representations are often *misaligned*. Without an understanding of personal space, the assistive robot moves the user's cup inches away from their face [43]. Without knowing about jaywalking pedestrians, the autonomous car cannot properly detect them and fatally crashes into them [202]. Without recognizing that the pilot is not aware of a new design feature, the flight control system enters a tug of war, nose diving into the ocean [105].

In this dissertation, we advocate that solving interaction starts with *aligning robot representations with humans*. Typical methods attempt to learn the human's representation of what matters in a task from aggregated data sets of human behavior, but this data will also not reflect every individual, environment, and task the robot will be exposed to. Instead, we suggest that we should treat humans as active participants in the interaction not as static data sources. Since humans have personalized representations of their tasks, they are uniquely equipped to communicate to the robot what they care about. This leads to the core insight of this dissertation:

> *successful human-robot interaction requires robots to engage with humans in an interactive process for finding a shared representation of what's salient for the task.*

Overall, this thesis approaches interaction by explicitly tackling representation alignment, rather than hoping it will naturally emerge. This lets us unify cognitive models of human behavior with machine learning methods in a way that's scalable, accessible, and robust, and develop practical algorithms for reliable and transparent human-aligned interaction. Throughout this dissertation we evaluate our methods in robot experiments with real human participants. Our key contributions are primarily demonstrated in interaction contexts like assistive and personal robotics, but our insights have broad implications to other application areas like self-driving, aircraft flight control systems, delivery quadcopters, and semi-autonomous surgical robots.

Fig. 1.1 provides a visual overview of the four main parts of the thesis (which we briefly motivate below) and how they fit in the broader space of representation alignment in human-robot interaction. We conclude this dissertation with a discussion outlining a few future research directions towards more seamless human-robot interaction.

Part I: Formalizing Representation Alignment. We have motivated that building robots that act for and with people necessitates representations that are not just functional but also reflective of what humans care about, i.e. that are *aligned*. However, it remains unclear how to study, analyze, or innovate upon representation alignment in the broader robotics and robot learning fields. In this first part, we present a unifying lens on aligning robot and human representations that serves as the foundation on top of which the remaining three parts of the thesis are built. In Chapter 2, we first identify the four key desiderata for the idealized human-aligned representation: recoverability, minimality, ease of recovery, and explainability. Based on these, we then mathematically operationalize the representation alignment problem in robotics. This allows us to situate current methods within this unified formalism, identify their strengths and weaknesses, and draw key takeaways to help discern directions for future work. We survey over 100 works on learning representations in robotics, and categorize, compare, and contrast them under this lens.

Part II: Learning Representations from Representation-Specific Input. Representation alignment demands a mechanism by which robots can quickly and efficiently learn, expand, or modify their representations from human input whenever necessary. The standard data-driven mechanism uses task-specific data sets of how people solve their tasks in hopes of *implicitly* recovering representations compatible with what humans want. Instead, here we *explicitly* include humans in the alignment process and directly ask them for representation-specific input to teach the robot about their representations. Chapter 3 proposes learning the human's representation *one feature at a time*, allowing non-expert users to hone into every feature. To reduce the amount of data necessary for teaching a feature, we maximize informativeness while minimizing human teaching burden by introducing a new type of representation-specific input: feature traces – trajectories that exhibit a monotonic evolution of the human's internal feature value. By exploiting this monotonicity property, we can efficiently train each feature from just a few traces. Chapter 4 provides an extension for teaching *perceptual features* – features that map from high-dimensional sensor input spaces, e.g. images or point clouds. Under this setting, learning a perceptual feature that is robust across the input space is very data-inefficient with standard supervised methods. To address that, we bootstrap the representation learning process by first training a proxy feature that operates on a low-dimensional transformation of the sensor space (e.g. geometries), and then treating it as an automatic labeler for a large and diverse set of high-dimensional observations in simulation. By breaking up the training process in this way, the robot can now learn perceptual features with much less data.

So far, we assumed that the human can identify every individual feature of the representation they wish to teach, but that's not always easy. Imagine you want the robot to understand what comfort means or learn a representation for emotion. How would you even explicate such a feature representation? In Chapter 5, rather than thinking of and teaching each feature one at a time, we observe that cognitive science grounds certain representations in well-studied *structures*, for which we can then ask the human for labels. We use as motivation generating robot motion that is expressive and we focus on one such structure that centers around representing emotion. By defining a known feature structure for emotion, the robot only has to learn how to map various of its motions to it, which is something a person can feasibly provide labels for. We show that, when trained this way, a variety of robots can now produce a vast array of expressive motions, from happy skipping to fearful tip toeing. Unfortunately, though, most human representations do not have extensive prior research on known feature structures. As such, in Chapter 6 we try to learn the human's representation *all features at once* by designing representation-specific tasks – proxy tasks intended for learning an embedding of everything that matters from the person's behavior on that task. We investigate a behavior similarity task where people decide whether robot behaviors are similar to one another. The intuition is that if a person decides two robot behaviors are similar, they must also internally represent them similarly, enabling us to extract their representation. With our approach, we recover representations that are much closer to the human's than the state-of-the-art, which we also show leads to more generalizable reward learning.

Part III: Detecting Representation Misalignment. Representation alignment has important implications for learning the right human-intended tasks. With the wrong representation, the robot may incorrectly anticipate what the person will do or misinterpret their guidance for how to do the task, resulting in undesired behaviors and poor coordination. For instance, if a manipulation robot does not know about the user's personal space, it may misunderstand their push to stay away as a correction for something else like holding mugs at an angle, accidentally causing a spill. In this third part, we enable the robot to automatically detect when its representation is misaligned with the human's and fix it. Chapter 7 proposes that the robot should be introspective and estimate a *confidence* in how much to trust its representation: if the person's behavior seems inexplicable under the robot's current representation, then that representation must be misaligned. This gives the robot a fast and principled way to monitor misalignment, and it can even overcome it: once the robot knows its representation is wrong, it can re-learn or expand it to be aligned with the human's. We demonstrate that the robot can still learn from behavior it understands, but is now robust to misinterpretation and ultimately learns what humans want. While this chapter showcases this insight in learning from physical demonstrations or corrections, Chapter 8 applies it even more broadly to teleoperation in shared autonomy. In that setting, the confidence parameter can additionally be used to arbitrate the degree of control the robot has during the task execution: if representations are aligned, higher confidence signals to the robot that it can take more of the control burden and assist the human user; however, if they are misaligned, low confidence naturally makes the robot relinquish control authority to the human so that it doesn't assist for the wrong goal.

Part IV: Using Aligned Representations to Interpret Human Behavior. Human behavior is not random – it is intentional and reflective of internal goals and preferences. To interpret and learn from it, robots rely on a *model* of how people act: they often assume humans trade off what matters for the task – the representation – and behave to achieve that trade-off. Once we agree that representations should not be just functional but also human-aligned, we can also fundamentally rethink how robots model people internally, and, thus in turn, interpret their behavior. In this final part, we propose a novel human behavior model that makes use of aligned representations. We observe that when humans make decisions, their representation affects how they view the options amongst which they are choosing. For example, although there are many paths around an obstacle, humans may group them into “left” and “right” when deciding between them. Instead of seeing human decisions as a choice from a set of behaviors, as current models do, our model reinterprets the available choices from the lens of the human-aligned representation. We demonstrate that this better reflects how humans make decisions, and that robots make better inferences if they interpret human input as such. Crucially, our model has wide-reaching impacts beyond robotics, to artificial intelligence, econometrics, and cognitive science.

Part I

Formalizing Representation Alignment

Part I serves as the formal foundation on top of which the rest of the thesis is built. In this dissertation, we advocate that robots that act for and with people necessitate representations that are not just functional but also reflective of what those people care about, i.e. that are aligned. Unfortunately, current learning approaches suffer from representation misalignment, where the robot's learned representation does not capture the human's representation. We suggest that because humans are the ultimate evaluator of robot performance, it is critical that we explicitly focus our efforts on aligning learned representations with humans, *in addition to* learning the downstream task. We advocate that representation learning approaches in robotics should be studied from the perspective of how well they accomplish the objective of representation alignment. We mathematically operationalize the representation alignment problem, identify its key desiderata, and situate current methods within this formalism.

Chapter 2

Aligning Robot and Human Representations

This chapter is based on the paper "Aligning Robot and Human Representations" [39], written in collaboration with Andi Peng, Pulkit Agrawal, Julie Shah, and Anca Dragan.

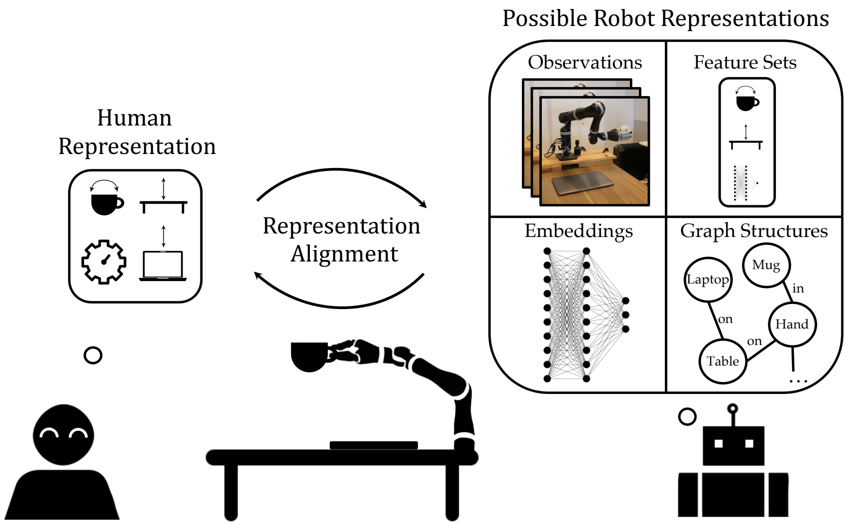


Figure 2.1: We formalize representation alignment as the search for a robot representation that is *easily able* to capture the true human task representation. We review four categories of current robot representations.

In robot learning, we aspire to build robots that perform tasks that human users want them to perform. To do so, robots need good *representations* of salient task features. For example, in Fig. 2.1, to carry a coffee mug, the robot considers efficiency, mug orientation, and distance from the user's possessions in its behavior. There are two paradigms for learning representations: one that *explicitly* builds in structure for learning features, e.g. feature sets or graphs, and one that *implicitly* extracts features by mapping input directly to desired behavior, e.g. end-to-end approaches [179, 244]. While explicit structure is useful for capturing relevant task features, it's often impossible to comprehensively define all features that may matter to the downstream task; meanwhile, implicit methods circumvent this problem by allowing neural networks to automatically extract representations, but they are prone to capturing *spurious correlations* [179], resulting in potentially arbitrarily bad robot behavior under distribution shift between train and test conditions [223].

Our observation is that many failures in robot learning, including the ones above, often result from *a mismatch* between the human's representation and the one learned by the robot; in other words, their representations are *misaligned*. From this perspective, these failures illuminate that if we truly wish to learn good representations – if we truly want robots that do what human users want – we must explicitly focus on the foundational problem: *aligning robot and human representations*. We offer a unifying lens for the robot learning community to view existing and future solutions to this problem.

We review over 100 papers in the representation learning literature in robotics from this perspective. We first define a unifying mathematical objective for an aligned representation based on four desiderata: value alignment, generalizable task performance, reduced human burden, and explainability. We then conduct an in-depth review of four common representations (Fig. 2.1): the identity representation, feature sets, feature embeddings, and graphical structures – illustrating the deltas each falls short in with respect to the desiderata. From situating each representation in our formalism, we arrive at the following key takeaway: a better structured representation affords better alignment and therefore better task performance, but always with the unavoidable tradeoff of more human effort. This effort can be directed in three ways: 1) representations that operate directly on the observation space, e.g. end-to-end methods, direct effort at increasing task data to avoid spurious correlations; 2) representations that build explicit task structure, e.g. graphs or feature sets, direct effort at constructing and expanding the representation; and 3) representations that learn directly from implicit human representations, e.g. self-supervised models, direct effort at creating good proxy tasks.

This chapter is much like a survey, except there is little work that directly addresses the representation alignment problem we pose. Instead, we offer a retrospective on works that focus on learning task representations in robotics with respect to our desiderata. Our review provides a unifying lens to think about the current gaps present in the robot learning literature as defined by a common language, or in other words, a roadmap for thinking about challenges present in current and future solutions in a principled way.

2.1 The Desired Representation

We start off by building intuition for the desiderata defining aligned representations.

Value Alignment. Learning human-aligned representations can aid with *value alignment* [14], enabling robots to perform well under the human's desired objective rather than optimize misspecified objectives that lead to unintended side-effects. In "reward hacking" scenarios, the reward function may capture an ill-defined or misleading representation of the human's intent [14]. In the canonical example of a robot tasked with sweeping dust off the floor [248], an optimal policy for the reward "maximize dust collected off the floor" leads the robot to dump dust just to immediately sweep it up again. In this case, the reward is defined on top of a representation that is *under-specified*, e.g. the amount of dust that is collected, and fails to capture other important features, e.g. covering the whole house, not adding dust on the floor, etc. Explicitly learning a representation that is aligned to the human's could ensure that causal features for accomplishing the desired objective are fully captured.

Generalizable Task Learning. A human-aligned representation may afford more generalizable task learning [91, 20]. A central problem in robot learning is our ability to learn diverse behaviours across different environments and user preferences [179, 223]. While domains like natural language or vision have elicited impressive performance across many tasks by leveraging large-scale datasets [234, 53, 229], robot learning is bottlenecked by our ability to collect diverse data that captures the complexity of the world. Without it, neural networks may learn non-causal correlates in the input space [143, 121]. Thus, learning objectives that operate directly on high-dimensional input spaces suffer from *spurious correlations*, where the implicit representation may contain features that are irrelevant to the task [8]. Consequently, the learned network may be based on these correlated irrelevant features that appear causal in-distribution, but fail under distribution shift. Explicitly aligning robot representations with those used by humans may afford more generalizable and robust task learning under distribution shift.

Reducing Human Burden. Operating on human-aligned representations may reduce teaching burden. In our above scenarios where human guidance is either task demonstrations or specified rewards, if we had unlimited human time and effort, we would be able to provide a perfect task representation, i.e. a demonstration of the task in every environment for every user [94], or a reward function that specifies every feature any user may find relevant for performing the task in any environment [124], and then fit the data with an arbitrarily complex function such as a neural network. In practice, both scenarios are computationally intractable with low sample complexity, and therefore motivate the need for representations that align with humans on the task abstraction level [3, 137, 4].

Explainability. We want representations that enable system transparency for ethical, legal, safety, or usability reasons [108, 11]. Current methods range from generating post-hoc explanations [108, 33], text descriptions of relational MDPs [130, 253], or saliency maps [113] for explaining behavior. However, system interpretability should not only be considered during deployment, but also be embedded within the design process itself [104, 116]. Explicitly aligning representations with humans' can create a more streamlined process for ensuring that representations are primed for human understanding [245].

Desideratum 1: The representation should capture *all* relevant task features, i.e., the human's true objective should be *realizable* when using the representation for task learning.

Desideratum 2: The representation should not capture *irrelevant* features of the desired task, i.e., the representation should not be based on spurious correlations.

Desideratum 3: Human guidance for learning the representation should demand *minimal* time and effort, i.e., the human's representation should be *easily recoverable* from data.

Desideratum 4: The representation should enable system *interpretability* and *explainability*, affording safe, transparent systems that can integrate with human users in the real world.

We henceforth refer to these desiderata as D1-4, mathematically operationalize them in the context of learning robot representations from human input, and situate how prior works relate to these goals.

2.2 The Representation Alignment Problem Formulation

Setup. We consider cases where a robot R seeks to learn how to perform a task desired by a human H. The two agents live in state $s \in \mathcal{S}$ and execute actions $a_H \in \mathcal{A}_H$ and $a_R \in \mathcal{A}_R$. The robot's goal is to learn a task expressed via a *reward function* $r^* : \mathcal{S} \to \mathbb{R}$ capturing the human's preference over states. The human knows the desired task, and, thus, implicitly knows r^* and how to act accordingly via a *policy* $\pi^*(a_H \mid s) \in [0, 1]$, but the robot does not and has to learn that from the human.

We consider two popular robot learning approaches: *imitation learning*, where we learn the human's policy for solving the task, and *reward learning*, where we learn the reward function describing the task. The approaches have different trade-offs: imitation learning does not require modeling the human and simply replicates their actions [219, 2], but in doing so it also replicates their suboptimality and can't generalize well to changing dynamics or state distributions [179, 276]; meanwhile, reward learning attempts to capture *why* a specific behaviour is desirable and, thus, can generalize better to novel scenarios [2] but requires assuming a human model and large amounts of data [102, 241].

Partial Observability and Representations. In theory, the state s could comprehensively capture the "true" components of the world down to their atomic elements, but in practice such a hypothetical state is neither fully observable nor useful. Instead, we assume that neither agent has the full state but they each *observe* it via observations $o_H \in \mathcal{O}_H$ and $o_R \in \mathcal{O}_R$. The robot's observations o_R come from its (possibly noisy and non-deterministic) sensors $P(o_R \mid s)$, e.g. robot joint angles, RGB-D images, object poses and bounding boxes, etc. The human also senses observations o_H via their "sensors", e.g. retinal inputs, audio signals, etc., which we could model according to $P(o_H \mid s)$. Due to partial observability, both the robot and the human use the *history* of t observations $\mathbf{o}_R = (o_R^1, ..., o_R^t) \in \mathcal{O}_R^t$ and $\mathbf{o}_H = (o_H^1, ..., o_H^t) \in \mathcal{O}_H^t$, respectively, as a proxy for the state – or sequence of states – they observe $\mathbf{s} = (s^1, ..., s^t) \in \mathcal{S}^t$. We assume that $\mathbf{o}_R$ and $\mathbf{o}_H$ correspond to the same $\mathbf{s}$.

Literature suggests that humans don't estimate the state directly from the complete $\mathbf{o}_H$ [35]. Instead, people focus on what's important for their task, often ignoring task-irrelevant attributes [59], and build a task-relevant *representation* to help them solve the task [46]. We, thus, assume that when humans think about how to complete or evaluate a task, they operate on a representation $\phi_H(\mathbf{o}_H)$ given by the transformation $\phi_H : O_H^t \rightarrow \Phi_H$, which determines which information in $\mathbf{o}_H$ to focus on and how to combine it into something useful for the task. For example, to determine if two novel objects have the same shape, a human might first look around both of them (gather a sequence of visual information $\mathbf{o}_H$) to build an approximate 3D model (representation $\phi_H(\mathbf{o}_H)$). Intuitively, we can think of such a representation as an estimate of the task-relevant components of the state, in lieu of the true unknown state. We can, thus, model the human as approximating their preference ordering r^* with a reward function $r_H : \Phi_H \rightarrow \mathbb{R}$, and their policy mapping π^* with $\pi_H(a_H \mid \phi_H(\mathbf{o}_H)) \in [0, 1]$.

The robot can similarly hold representations $\phi_R(\mathbf{o}_R)$ given by $\phi_R : O_R^t \rightarrow \Phi_R$. The most general ϕ_R is the identity function, where the robot uses the raw observations directly, but Sec. 2.4 will also inspect more structured representations. For example, representations can be instantiated as handcrafted feature sets, where the designer distills their prior knowledge by pre-defining a set of representative aspects of the task [25, 214, 123], or as neural network embeddings, where the network tries to implicitly extract such prior knowledge from data demonstrating how to do the task [96, 257, 296].

Imitation Learning. Here, the robot's goal is to learn a policy π_R that maps from its representation to a distribution over actions $\pi_R(a_R \mid \phi_R(\mathbf{o}_R))$ telling it how to successfully complete the task. To do so, the robot receives task *demonstrations* from the human and learns to imitate the actions they take at every state [219, 276]. Let the human demonstration be a state trajectory $\xi = (s^0, \ldots, s^T)$ of length T. Importantly, the human and the robot perceive this trajectory differently: the human observes $\xi_H = (o_H^0, \ldots, o_H^T)$ and the robot $\xi_R = (o_R^0, \ldots, o_R^T)$. Because the demonstrator is assumed to produce trajectories with high reward $r_H(\phi_H(\xi_H))$, i.e. be a task expert, the intuition is that directly imitating their actions should result in good behaviour without the need to know the reward.

The issue with this approach is that the human's policy $\pi_H(a_H \mid \phi_H(\mathbf{o}_H))$ produces actions based on $\phi_H(\mathbf{o}_H)$, whereas the robot's actions are based on $\phi_R(\mathbf{o}_R)$. By directly imitating the human, the method, thus, implicitly assumes that $\phi_H(\mathbf{o}_H)$ is accurately captured by – or easily recoverable from – whatever $\phi_R(\mathbf{o}_R)$ was chosen to be. In other words, it assumes the robot and human's representations of what matters for the task are naturally *aligned*. If this assumption does not hold, the robot might not recover the right policy, and, thus, execute the right actions at the right state.

Reward Learning. Here, the robot's goal is to recover a parameterized estimate of the human's reward function $r_\theta : \Phi_R \rightarrow \mathbb{R}$, from either demonstrations [311, 96], corrections [25], teleoperation [145], comparisons [71], trajectory rankings [50] etc. The intuition here is that the human's input can be interpreted as evidence for their internal reward function r_H, and the robot can use this evidence to find its own approximation of their reward r_θ. Given a learned r_θ, the robot can find an optimal policy π_R by maximizing the expected total reward $\mathbb{E}_{\pi_R}[\sum_{t=0}^{\infty} r_\theta(\phi_R(\mathbf{o}_R))]$.

Similar to imitation, because the human internally evaluates the reward function r_H

based on $\phi_H(\mathbf{o}_H)$, their input is also based on $\phi_H(\mathbf{o}_H)$, whereas the robot interprets it as if it were based on $\phi_R(\mathbf{o}_R)$. Hence, if the two representations $\phi_R(\mathbf{o}_R)$ and $\phi_H(\mathbf{o}_H)$ are *misaligned*, the robot may recover the wrong reward function, and, thus, produce the wrong behaviour when optimizing it [38, 100].

The Problem of Misaligned Representations. In this thesis, we reflect on the traditional assumptions that robot learning are built on and encourage not taking representation alignment for granted:

In the real-world, we cannot assume that robot and human representations will naturally align.

We see this in our examples of robot representations $\phi_R(\mathbf{o}_R)$. The identity "representation" which maps $\mathbf{o}_R$ onto itself should, in theory, capture everything in $\phi_H(\mathbf{o}_H)$ so long as $\mathbf{o}_R$ has enough information, but the high-dimensionality of $\mathcal{O}_R^t$ makes this representation impractical: learning a reward or policy that is robust across the input space and generalizes across environments would require a massive amount of diverse data – an expensive ask when working with humans [241, 102]. A set of feature functions is lower dimensional, but pre-specifying all features that may matter to the human is unrealistic, inevitably leading to representations $\phi_R(\mathbf{o}_R)$ that lack aspects in $\phi_H(\mathbf{o}_H)$ [38]. Learning neural network embeddings $\phi_R(\mathbf{o}_R)$ that map from the history $\mathbf{o}_R$ while robustly and generalizably covering all $\mathbf{o}_R$ (and, thus, $\mathbf{o}_H$) requires a lot of highly diverse data, similar to how reward and policy learning on the identity representation would. In summary, whether it's insufficient knowledge of what matters for the task or insufficient resources for exhaustively demonstrating the task, the robot's representation will more often than not be misaligned with the human's.

2.3 A Formalism for Representation Alignment

How can we mathematically operationalize representation alignment? While it is impossible for the robot and the human to perceive the world the same via $\mathbf{o}_R$ and $\mathbf{o}_H$, in an ideal world we would want them to *make sense of their observations in a similar way*. To that end, we formalize the *representation alignment problem* as the search for a robot representation that is similar to the human's representation. This takes the form of an optimization problem with the following objective:

$$\phi_R^* = \arg\max_{\phi_R} \psi(\phi_R, \phi_H), \tag{2.1}$$

where ψ is a function that measures the similarity (or alignment) between two representation functions. The question is how do we measure representation alignment, i.e. what is ψ? We pose the ψ below:

$$\psi(\phi_R, \phi_H) = -\min_F \sum_{\mathbf{s}\in\mathcal{S}^t} \|F^T \phi_R(\mathbf{o}_R) - \phi_H(\mathbf{o}_H)\|_2^2 - \lambda \cdot \dim(\Phi_R) \ , \tag{2.2}$$

where $\mathbf{o}_R$ and $\mathbf{o}_H$ correspond to $\mathbf{s}$, F is a linear transformation, and λ is a trade-off term. We next further explain this notation and why (2.2) best reflects our desiderata from Sec. 2.1.

D1: Recover the Human's Representation. To ensure the robot's representation captures *all* relevant task aspects, we intuitively want alignment to be high when the human's representation can be *recovered* from the robot's, no matter the state(s) $\mathbf{s}$. Mathematically, we define "recovery" as a mapping $f : \Phi_R \to \Phi_H$ from $\phi_R(\mathbf{o}_R)$ to $\phi_H(\mathbf{o}_H)$, where $\phi_H(\mathbf{o}_H)$ is recoverable from $\phi_R(\mathbf{o}_R)$ if $f(\phi_R(\mathbf{o}_R)) \approx \phi_H(\mathbf{o}_H), \forall \mathbf{s}$, where $\mathbf{o}_R$ and $\mathbf{o}_H$ correspond to $\mathbf{s}$. In other words, we can express the recovery error via an L_2 distance summed across all state sequences $\mathbf{s}$: $\sum_{\mathbf{s} \in \mathcal{S}^t} \|f(\phi_R(\mathbf{o}_R)) - \phi_H(\mathbf{o}_H)\|_2^2$. In (2.2), we want representation functions ϕ_R that have high alignment ψ with ϕ_H to have low recovery error, hence we use the negative best distance as a measure of similarity.

D2: Avoid Spurious Correlations. We want $\phi_R(\mathbf{o}_R)$ to not just recover $\phi_H(\mathbf{o}_H)$, i.e. be sufficient, but also be *minimal* to avoid spurious correlations that reflect irrelevant task aspects. We formalize this with a penalty on the dimensionality of the robot representation function's co-domain Φ_R. Together, **D1** and **D2** describe in (2.2) a measure of representation alignment that rewards small representations that can be mapped close to $\phi_H(\mathbf{o}_H)$, where λ is a designer-specified trade-off parameter.

D3: Easily Recover the Human's Representation. We operationalize the ability to *easily* recover $\phi_H(\mathbf{o}_H)$ from $\phi_R(\mathbf{o}_R)$. Finding an optimal solution to (2.2) via typical optimization methods is intractable given the large space of functions f to search over. In theory, if the human's ϕ_H can be queried by the robot (e.g., by asking for labels), the most straightforward solution collects feedback $\langle \mathbf{o}_R, \phi_H(\mathbf{o}_H) \rangle$ from the human and fits an approximation $\hat{f}(\phi_R(\mathbf{o}_R)) \approx \phi_H(\mathbf{o}_H)$, e.g. a neural network. Unfortunately, even if $\phi_R(\mathbf{o}_R)$ is low-dimensional, fitting an arbitrarily complex $\hat{f}$ that reliably results in high alignment for all states could require a large amount of representative labels, i.e. it would not be *easy* to recover the human's representation. For this reason, we want "easy" recovery to involve a transformation f of small complexity. This condition has been mathematically stated via a multitude of complexity theory arguments (upper bounds based on the Vapnik–Chervonenkis dimensions [31, 127, 29, 153] or the Radamacher complexity of the function [111, 30]), but recent empirical work argues that linear transformations are a good proxy for small complexity [73, 169, 243, 10]. We thus similarly take f to be a linear transformation given by a matrix F.

D4: Explain the Robot's Representation. Human-aligned representations should be amenable to interpretability and explainability tools. If the human representation is easily recoverable, i.e. the robot can learn a good estimate $\hat{f}$, we get this condition almost for free: the robot can communicate its representation to the human by showing examples $\langle \mathbf{o}_H, \hat{f}(\phi_R(\mathbf{o}_R)) \rangle$ where observation sequences are labeled with the robot's current "translation" of its representation. The last piece we need for explainability is ensuring that $\hat{f}$ is understandable by the human, by, for example, having additional tools that can convert $\hat{f}$ into more human-interpretable interfaces, like language or visualizations.

Examples of Robot Representations. Since solving (2.1) is intractable for an arbitrarily large set of functions ϕ_R, different ways of defining the robot's representation $\phi_R(\mathbf{o}_R)$ implicitly make different simplifying assumptions. When ϕ_R is the identity function, the underlying assumption is that there exists some $f : O_R^t \rightarrow \Phi_H$ that satisfies (2.2) so long as $\mathbf{o}_R$ has enough information to capture $\phi_H(\mathbf{o}_H)$. Unfortunately, because f operates on an extremely large space of robot observation histories O_R^t, it would have to be complex enough to reliably cover the space, violating **D3**. This, together with the large dimensionality of the representation space, result in a small alignment value in (2.2). Meanwhile, methods that assume that $\phi_R(\mathbf{o}_R)$ has some more low-dimensional structure, like the feature sets or embeddings from earlier, could also have small alignment values: feature sets might be non-comprehensive, while learned feature embeddings might have not extracted what's truly important to the human, making it, thus, impossible to find an f that recovers $\phi_H(\mathbf{o}_H)$. As we will see in Sec. 2.4, no representation is naturally human-aligned and every representation type comes with its trade-offs.

Extension to Multiple Tasks. We have considered the single task setting, where the robot's goal is to successfully perform one desired task, but our formalism can be extended to account for multiple tasks. First, when the person wants to train the robot to correctly perform multiple tasks, the observation space O_R may be different for each task. In practice, these observation spaces are oftentimes the same or similar (e.g. multiple robot manipulation tasks can all still use images of the same tabletop as observations, although the observation distribution may differ if different objects are used). We can account for differing spaces by choosing the overall observation space O_R to be the union of all individual N task observation spaces O_{R_i}: $O_R = O_{R_1} \bigcup \ldots \bigcup O_{R_N}$. Additionally, in multi-task settings, the human representation $\phi_H(\mathbf{o}_H)$ will reflect aspects of the task *distribution* that matter to them, rather than of a single task.

Extension to Multiple Humans. Aligning the robot's representation to multiple humans requires acknowledging that each human may operate under a different observation space O_H or representation $\phi_H(\mathbf{o}_H)$. First, we could modify our formalism for differing spaces similarly to how we did in the multi-task setting, by choosing the overall observation space O_H to be the union of all individual M human observation spaces O_{H_i}: $O_H = O_{H_1} \bigcup \ldots \bigcup O_{H_M}$. Second, in such multi-agent settings, the robot could attempt to align its representation to a unified $\phi_H(\mathbf{o}_H) = \phi_{H_1}(\mathbf{o}_H) \bigcup \ldots \bigcup \phi_{H_M}(\mathbf{o}_H)$, individually to each $\phi_{H_i}(\mathbf{o}_H)$, or a combination of the two strategies where the unified representation is then specialized to each individual human's representation.

2.4 Survey of Robot Representations

We present our survey on *learned* robot representations. Table 2.1 situates them within our formalism's key tradeoffs. We focus on 4 categories of representations: the identity map, feature sets, feature embeddings, and graph structures.

Table 2.1: Existing representations (and example papers) through the lens of our formalized desiderata.

Representation Type	**D1: Recoverability of $\phi_H(\mathbf{o}_H)$ from $\phi_R(\mathbf{o}_R)$** $\min_f \sum_{\mathbf{s}\in\mathcal{S}^t} \lVert f(\phi_R(\mathbf{o}_R)) - \phi_H(\mathbf{o}_H)\rVert_2^2$	**D2: Minimality** $\dim(\Phi_R)$	**D3: Ease of Recovery of $\phi_H(\mathbf{o}_H)$ from $\phi_R(\mathbf{o}_R)$** $\min_F \sum_{\mathbf{s}\in\mathcal{S}^t} \lVert F^T\phi_R(\mathbf{o}_R) - \phi_H(\mathbf{o}_H)\rVert_2^2$	**D4: Interpretability**
Identity $\phi_R(\mathbf{o}_R) = \mathbf{o}_R \in \mathcal{O}_R^t$ [219, 276, 96, 71, 95, 296]	Contains complete information	$\lvert\mathcal{O}_R^t\rvert$, Large	Difficult in arbitrarily large observation spaces	Black box
Feature Set $\phi_R(\mathbf{o}_R) = \{\phi_R^1(\mathbf{o}_R), \ldots, \phi_R^d(\mathbf{o}_R)\}$ [281, 68, 177, 235, 40, 42, 224, 303]	May lack information but can use misalignment detection methods to learn new features	d, Grows linearly	If complete, easy If complete but d large, medium If incomplete, hard	High
Feature Embedding (Unsupervised) $\phi_R(\mathbf{o}_R) = \vec{\phi}_R(\mathbf{o}_R) \in \mathbb{R}^d$ [174, 269, 256, 15, 305, 119, 170, 125]	May learn wrong disentangled information	d, Low by design	If disentangled information complete, easy If disentangled information incomplete, hard	May be interpretable to designer
Feature Embedding (Supervised) $\phi_R(\mathbf{o}_R) = \vec{\phi}_R(\mathbf{o}_R) \in \mathbb{R}^d$ [50, 278, 109, 239, 135, 45, 140]	More likely to capture relevant information	d, Low by design	If relevant information, easy If missing relevant information, hard	May be interpretable to designer
Graph $\phi_R(\mathbf{o}_R) = G = \{V, E\}$ [285, 304, 77, 301, 63, 207, 120, 307]	May lack information	$\lvert V+E\rvert$, $\lvert V\rvert$ linear, $\lvert E\rvert$ quadratic	If complete, easy If complete but $\lvert V+E\rvert$ large, medium If incomplete, hard	High

2.4.1 Identity Representation

The identity maps an observation history onto itself, i.e. $\phi_R(\mathbf{o}_R) = \mathbf{o}_R$ with $\phi_R : \mathcal{O}_R^t \to \mathcal{O}_R^t$. The methods we review here, thus, don't learn an explicit intermediate representation but instead hope to implicitly extract what's important from human task data.

Because the inputs for reward or policy learning consist of potentially high-dimensional observation histories, e.g. images, most approaches we cover here are based on high-capacity deep learning models. There are now numerous end-to-end methods for learning policies [219, 276, 241, 179] or rewards [96, 102, 291] from demonstrations. While these methods perform well with an overparameterized high complexity function, they tend to overfit to the training tasks and suffer from generalization failures due to *distribution shift* [244], resulting in arbitrarily erroneous behaviour during deployment. Achieving good end-to-end performance across a large test distribution can require hundreds or even thousands of demonstrations for each desired task [306, 230, 231], which is expensive to obtain in practice. In reward learning, this has been alleviated by introducing other types of human reward feedback, like comparisons [71], numeric feedback [286], examples of what is a goal [103], or a combination [142]. These are not only more user friendly alternatives to demonstrations, but they are also amenable to active learning techniques [242, 263], which can further reduce the human burden.

Another popular way to reduce the sample complexity is meta-learning [95], which seeks to learn representations that can be quickly fine-tuned [296, 302, 264, 141, 257]. The intuition is that we can reuse human data from many different tasks, and if the training distribution is representative enough, this "warm-started" model can adapt to new tasks with little data. Unfortunately, the human needs to know the test task distribution *a priori*, which brings us back to the specification problem: we now trade hand-crafting features for hand-crafting task distributions. Moreover, because these models are overparameterized, they are inherently *uninterpretable* and tough to debug in case of failure [245].

Takeaway. Despite recent advances in end-to-end systems, the identity representation, while easy to specify, is difficult to use for robust and generalizable robot learning while minimizing human input.

2.4.2 Feature Sets

We can instantiate $\phi_R(\mathbf{o}_R)$ as a set $\{\phi_R^1(\mathbf{o}_R), ..., \phi_R^d(\mathbf{o}_R)\}$, where each $\phi_R^i(\mathbf{o}_R)$ is a different individual dimension of the representation, with d much smaller than $|\mathcal{O}_R^t|$. These dimensions represent concrete task aspects – or features, e.g. how far the end effector is from the table, – which is why we call ϕ_R^i a feature function and the output $\phi_R^i(\mathbf{o}_R)$ a feature value. The feature function maps observation histories to a real number indicating how much that feature is expressed in the observations, $\phi_R^i : \mathcal{O}_R^t \to \mathbb{R}$. The robot's representation maps from observation histories onto a d-dimensional space of real values: $\phi_R : \mathcal{O}_R^t \to \mathbb{R}^d$, where d grows linearly with the number of features.

Handcrafted feature sets have been used widely across policy and reward learning [2, 144, 145, 258], but exhaustively pre-specifying *everything* a human may care about is impossible [43]. To address this, early methods infer relevant feature functions directly from task demonstrations. Vernaza and Bagnell [281] define the robot's representation as the PCA components of the observations, while other methods specify base feature components for constructing the feature functions [68] as either logical conjunctions [177] or regression trees [235]. Unfortunately, relying on engineering a relevant set of base features can be tedious and incomplete. Therefore, more recent methods instead learn individual feature functions as neural networks [40, 41, 42, 224, 303], and train them with labels for different diverse observations [224, 303]. While Paxton et al. [224] can learn complex spatial relations mapping from high-dimensional point cloud observations, they require large amounts of data, making the approach unsuitable for learning many different feature functions from a human. A different approach reduces the data complexity by introducing a new type of structured input, a feature trace, which results in large amounts of feature value comparisons to train the network with little effort from the human [40, 41]. Bobu et al. [42] reduce the burden by using a small amount of human labels to learn feature functions defined on a lower dimensional transformation of the observation space (object geometries) and using that to label data in a simulator (object point clouds).

Takeaway. While feature sets are advantageous for inserting structure in the downstream learning pipeline making it more data efficient, robust, and generalizable [41], that added structure is *only useful if complete*. Under-specified feature sets can be handled by detecting misalignment [38] and learning new features over time, but we need more ways to reduce the human burden for teaching features, like introducing new types of structured input [40] or bootstrapping the learning [42]. If, on the other hand, the structure is overcomplete, i.e. it contains irrelevant features, it may lead to spurious correlations which can be prevented via feature subset selection methods [58, 55, 196].

2.4.3 Feature Embeddings

We can instantiate $\phi_R(\mathbf{o}_R)$ as a low-dimensional feature embedding, or vector, $\vec{\phi}_R(\mathbf{o}_R)$, where each dimension is a different neuron in the embedding. The representation function is $\phi_R : \mathcal{O}_R^t \rightarrow \mathbb{R}^d$, with d fixed by the designer and much smaller than $|\mathcal{O}_R^t|$. While feature set functions also map to $\mathbb{R}^d$, each dimension is learned individually (and is representative of some task aspect), whereas here the embedding is learned jointly (and hopes to capture important task aspects implicitly). We identify two broad areas: unsupervised methods (also called self-supervised), which use unlabeled data and proxy tasks to learn representations, and supervised methods, which use human supervision at the representation level. We also cover some in-between semi- or weakly-supervised methods.

Unsupervised methods. At one extreme, unsupervised methods try to learn disentangled latent spaces from data collected without any human supervision. Instead of explicitly giving feedback, the human designer hopes to instill their intuition for what is causal for the task by specifying useful *proxy tasks* [174, 67, 171, 299]. In robot learning, these proxy tasks range from reconstructing the observation (to ignore irrelevant visual aspects) [97, 134, 119, 197], to predicting forward dynamics (to capture what constrains movement) [287, 119] or inverse dynamics (to recover actions from observations) [222], to enforcing behavioural similarity between observations [305, 106, 22], to contrastive losses [218, 170, 15, 269], or some combination [125, 256]. The proxy task result itself does not matter; rather, these methods are interested in the intermediate representation extracted from training on the proxy tasks. However, because they are purposefully designed to bypass supervision, these representations do not necessarily correspond to human features, rendering explicit alignment challenging. In fact, the cases where the disentangled factors match human concepts are primarily due to spurious correlations [193]. Lastly, like all learned latent representations, they are difficult to interpret and use to debug failures.

Supervised Methods. At the other extreme sit supervised approaches. Some methods combine the human's reward or policy data with self-supervised proxy tasks to pre-train a useful low-dimensional feature embedding [50, 278] while others reduce supervision by learning a simpler model that, when trained well, can automatically label large swaths of videos of people doing tasks [26]. Multi-task methods pre-train representations from human input for multiple tasks, then fine-tune the reward or policy on top of the learned embedding at test time [109, 215, 298]. Similar to meta-learning, the motivation here is that the robot collects data from many different but related tasks, which it can then leverage to jointly train a shared representation. This is more scalable than meta-learning [201], but still needs curating a large set of training tasks to cover the test distribution.

There is a growing body of work directly targeting supervision at the representation level. *Implicit* methods make use of a proxy task for the human to solve and a visual interface that changes based on the robot's current representation [239, 135, 45]. The hope is that if the human can still solve the proxy task well, the representation producing the visualization must contain causal behavioural aspects. If the representation dimensions are interpretable enough, *explicit* learning of aligned representations is also possible by directly labeling examples with the embedding vector values [140, 266]. What both these directions have in common is that the representation *is* or *can be* converted into a form that is interpretable to the human, thus opening the possibility of the human providing targeted feedback that is explicitly intended to teach the robot the desired task representation.

Takeaway. There is a trade-off between the amount of supervision at the representation level and how human-aligned the learned representations are. "Supervising" by coming up with proxy tasks certainly reduces the end user's labeling effort, but may result in misalignment. On the other hand, direct supervision more explicitly aligns the robot's representation with the human's, but is also more effortful. These representations tend to be more interpretable than the identity [93].

2.4.4 Graphical Structures

The fourth group maps observation histories onto a graph $G = \{V, E\}$, i.e. $\phi_R(\mathbf{o}_R) = G$ with $\phi_R : \mathcal{O}_R^t \to \mathcal{G}$. Many graphical structures have been used for robot learning and planning, from knowledge graphs [77], to directed graphs [254], Markov random fields [115], Bayesian networks [160], hierarchical task networks [207], etc. Graphical structures have been especially useful for fast and robust planning [21, 217, 185] or when robust robot behaviour relies on strong priors for the task context, like interpreting ambiguous user commands [304, 301] or handling partially observable environments [216, 77]. Since their relational structure directly allows for probing the causal effect of a certain part of the representation on the robot's behaviour [293, 76, 77], they are also often leveraged in the interpretability literature. Unfortunately, they require well-conceived, well-structured, and comprehensive domain knowledge to be successful [213, 206, 265, 80], which takes a considerable amount of human effort or data to build or learn [285, 187].

Takeaway. While graphical structures are more interpretable to users, they require significant human effort to construct and maintain relative to their neural network counterparts. Much like specifying rewards by hand, it is hard to specify all relevant nodes, potentially resulting in under-specification.

2.5 Discussion

We proposed a formal lens for viewing *representation alignment* in robot learning. While we do not offer a practical solution for (2.2), we believe there is still tremendous value in explicitly formalizing representation alignment. The formalism allows us to identify gaps in current methods (including the ones in Tab. 2.1) and provide directions for future work. Moreover, explicitly defining representation alignment in robotics as the optimization (2.2) allows us to assess future methods based on how well they approximate solutions to this problem. This thesis provides one way to tackle the representation alignment problem, but we hope future work will seek novel approximations to (2.2) to fill in the gaps.

Part II

Learning Representations from Representation-Specific Input

We have motivated that when robots interact with people, they need a representation capturing salient features of human decision-making, e.g. humans care how close to them the robot gets when carrying full mugs. Part II focuses on how robots can learn these representations from humans to ensure they are aligned. With standard data-driven methods, not only does the data used have to reflect every individual, environment, and task the robot will face, but it has to also implicitly extract the relevant task features. Instead of treating humans as static data sources, the core idea in Part II is to *explicitly include them in the alignment process and intelligently ask them to teach the robot about their representation*. This requires us to devise algorithms that make it easy for people of varying levels of expertise to give data about their representations. We do so in the context of learning every feature of the representation one at a time (Chapters 3 and 4), learning the mapping to a known feature structure (Chapter 5), and learning the representation all at once (Chapter 6). We *design representation-specific human input* that can still be leveraged by data-driven techniques but in a more efficient way (Chapter 3), we *bootstrap the representation learning process* to reduce the amount of data necessary (Chapter 4), we *structure the representation* to allow people to give feedback more easily (Chapter 5), and we *design representation-specific tasks* to reduce the person's cognitive load when teaching the robot (Chapter 6). By combining these techniques with the power of neural networks, we find that humans can more quickly teach the robot new features like the distance to their possessions, their personal space, or even emotion. We show that teaching representations has important benefits for reward learning, personalized motion planning, and expressive robot motion.

Chapter 3

One-by-One Representation Learning

This chapter is based on the papers "Feature Expansive Reward Learning: Rethinking Human Input" [40] and "Inducing Structure in Reward Learning by Learning Features" [41], written in collaboration with Marius Wiggert, Claire Tomlin, and Anca Dragan.

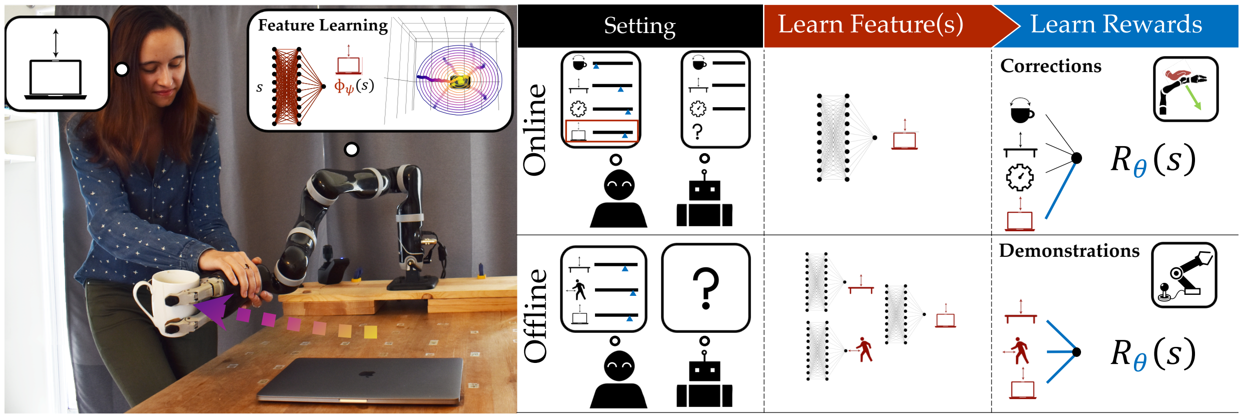


Figure 3.1: (Left) The person teaches the robot the feature for horizontal distance from the laptop by giving a few feature traces. (Right-Top) In the online reward learning from corrections setting, once the robot detects that its feature set is incomplete, it queries the human for feature traces that teach it the missing feature and adapts the reward to account for it. (Right-Bottom) In the offline reward learning from demonstrations setting, the person has to teach the robot each feature separately one at a time using feature traces, and only then teach their combined reward.

Whether it's semi-autonomous driving [249], recommender systems [311], or household robots working in close proximity with people [144], reward learning can greatly benefit autonomous agents to generate behaviors that adapt to new situations or human preferences. Under this framework, the robot uses the person's input to learn a reward function that describes how they prefer the task to be performed. For instance, in the scenario in Fig. 3.1, the human wants the robot to keep the cup away from the laptop to prevent spilling liquid over it; she may communicate this preference to the robot by providing a demonstration of the task or even by directly intervening during the robot's task execution to correct it. After learning the reward function, the robot can then optimize it to produce behaviors that better resemble what the person wants.

In order to correctly interpret and efficiently learn from human input, traditional methods resorted to structuring the reward as a (linear) function of carefully hand-engineered features – important aspects of the task [311, 2, 144, 25]. Unfortunately, selecting the right feature representation is notoriously challenging, even for expert system designers: knowing and specifying *a priori* an exhaustive set of *all* the features that might be relevant for the reward is impossible for most real-world tasks. To bypass this feature specification problem, state-of-the-art deep Inverse Reinforcement Learning (IRL) methods [291, 96, 50] learn rewards defined directly on the high-dimensional raw state (or observation) space, thereby implicitly constructing features automatically from task demonstrations.

In doing so, however, these approaches sacrifice the sample efficiency and generalizability that a well-specified feature set offers. While using an expressive function approximator to extract features and learn their reward combination at once seems advantageous, many such functions can induce policies that explain the demonstrations. Hence, to disambiguate between all these candidate functions, the robot requires a very large amount of (laborious to collect) data, and this data needs to be diverse enough to identify the true reward. For example, the human in the household robot setting in Fig. 3.1 might want to demonstrate keeping the cup away from the laptop, but from a single demonstration the robot could find many other explanations for the person's behavior: perhaps they always happened to keep the cup upright or they really like curved trajectories in general.

The underlying problem here is that demonstrations – or task-specific input more broadly – are meant to teach the robot about the reward and not about the features per se, so these function approximators struggle to capture the right feature representation for the reward. We argue that the robot does not have to learn everything at once; instead, it can *divide-and-conquer* the reward learning problem and focus on explicitly learning the features separately from learning how to combine them into the reward. In our earlier example, if the robot were taught about the concept of distances to laptops separately, it would be able to quickly tell what the person wants from a single demonstration.

We make the following contributions:

Learning features from a novel type of representation-specific human input. We present a method for learning complex non-linear features separately from the reward (Sec. 3.3). We introduce a new type of *representation-specific* human input specifically designed to teach features, which we call feature traces – partial trajectories that describe the monotonic evolution of the value of the feature to be learned. To provide a feature trace, the person guides the robot from states where the feature is highly expressed to states where it is not, in a monotonic fashion. Looking at Fig. 3.1 (Left), the person teaches the robot to avoid the laptop by giving a few feature traces: she starts with the arm above the laptop and moves it away until comfortable with the distance from the object. We present an algorithm that harvests the structure inherent to feature traces and uses it to efficiently learn a feature relevant for the reward: in our example, the horizontal distance from the laptop. In experiments on a 7-DoF JACO2 robot arm, we find that our method can learn high quality features closely resembling the ground truth (Sec. 3.5.1).

Demonstrating our feature learning in a user study on a simulated 7-DoF robot arm. In a user study with the JACO2 (Kinova) robotic arm, we show that non-expert users can use our approach for learning features (Sec. 3.5.2). The participants were able to provide feature traces to teach good features, and found our teaching protocol intuitive. Unfortunately, due to the current pandemic, we conducted the study online in a simulated environment; despite the inevitable degradation in input quality that this entails, the users were still able to teach features that induced informative bias.

Analyzing sample complexity benefits of learning feature representations for rewards. We show how our method, which we call Feature Expansive Reward Learning (FERL) because it expands the feature set one by one, can improve reward learning sample complexity and generalization. First, we look at an easier online reward learning setting like the one in Fig. 3.1 (Right-Top) where the robot knows part of the feature set from the get-go, but the person's preference also depends on other features not in the set (Sec. 3.4.2). We show that, by learning the missing feature, the robot obtains a more generalizable reward than if it had trained a deep IRL network directly from the raw state and the known set (Sec. 3.6). We then consider the more challenging offline reward learning case in Fig. 3.1 (Right-Bottom) where the person teaches the reward from scratch, one feature at a time (Sec. 3.4.1). We find that the robot outperforms the baseline most of the time, with less clear results when the learned features are noisily taught by novice users in simulation (Sec 3.7).

We show that taking a divide-and-conquer approach focusing on learning important features for the representation separately before learning the reward on top improves sample complexity in reward learning. Although showcased in manipulation, our method can be used in any robot learning scenarios where feature learning is beneficial: in collaborative manufacturing users might care about the rotation of the object handed over, or in autonomous driving passengers may care about how fast to drive through curves.

3.1 Prior Work

Programming robot behavior through human input is a well-established paradigm. In this paradigm, the robot receives human input and aims to infer a policy or reward function that captures the behavior the human wants the robot to express. In imitation learning, the robot directly learns a policy that imitates demonstrations given by the human [219]. The policy learns a *correlation* between situations and actions but not *why* a specific behavior is desirable. Because of that, imitation learning only works in the training regime whereas optimizing a learned reward, which captures *why* a behavior is desirable, can generalize to unseen situations [2].

In the IRL framework the robot receives demonstrations through teleoperation [145, 2] or kinesthetic teaching [17] and learns a reward under which these demonstrations are optimal [247, 2]. Recent research goes beyond demonstrations, utilizing other types of human input for reward learning such as corrections [144, 25], comparisons [71] and rankings [49], examples of what constitutes a goal [103], or even specified proxy objectives [123]. Depending on the interaction setting, the human input can be given all-at-once, iteratively, or on specific requests of the robot in an active learning setting [188, 51, 249].

All these methods require less human input if a parsimonious representation of the world, which summarizes raw state information in the form of relevant features, is available. This is because finite feature sets significantly reduce the space of possible functions which according to statistical learning theory reduces the information complexity of the learning problem [279]. In the following we discuss the the role of feature representations in reward learning and methods for learning features.

3.1.1 Feature Representations in Reward Learning

Traditional reward learning methods rely on a set of carefully hand-crafted features that capture aspects of the environment a person may care about. These are selected by the system designer prior to the task [311, 2, 144, 123, 25]. If chosen well, this feature set introduces an inductive bias that enables the algorithms to find a good estimate of the human's preferences with limited input. Unfortunately, selecting such a set in the first place is notoriously challenging, even for experts like system designers. For one, defining a good feature function can be a time consuming trial-and-error process, especially if the feature is meant to capture a complex aspect of the task [291]. Moreover, the chosen feature space may not be expressive enough to represent everything that a person might want (and is giving input about) [38, 128]. When this is the case, the system may misinterpret human guidance, perform unexpected or undesired behavior, and degrade in overall performance [13, 247, 128].

To tackle these challenges that come with hand-designing a feature set, state-of-the-art deep IRL methods use the raw state space directly and shift the burden of extracting behavior-relevant aspects of the environment onto the function approximator [96, 291]. The objective of IRL methods is to learn a reward which induces behavior that matches the state expectation of the demonstrations. The disadvantage of such approaches is that they require large amounts of highly diverse data to learn a reward function which generalizes across the state space. This is because with expressive function approximators there exists a large set of functions that could explain the human input, i.e. many reward functions induce policies that match the demonstrations' state expectation. The higher dimensional the state, the more human input is needed to disambiguate between those functions sufficiently to find a reward function which accurately captures human preferences and thereby generalizes to states not seen during training and not just replicates the demonstrations' state expectations. Thus, when venturing sufficiently far away from the demonstrations the learned reward in IRL does not generalize which can lead to unintended behavior [241, 102].

It has been shown that providing linear feature functions as human input can reduce the risk of unintended behavior [128]. In our work we argue that generalization with limited input can be achieved without requiring hand-crafted features if the robot explicitly learns features, instead of attempting to learn them implicitly from demonstrations.

3.1.2 Learning Features

In IRL researchers have explored the direction of inferring a set of relevant features directly from task demonstrations. This can take the form of joint Bayesian inference on both reward and feature parameters [68] or projecting the raw state space to lower dimensions via PCA on demonstrated trajectories [281]. There are also methods that add features iteratively to learn a non-linear reward, such as Levine, Popovic, and Koltun [177], which constructs logical conjunctions of primitive integer features, and Ratliff et al. [235], which trains regression trees to distinguish expert from non-expert trajectories in a base feature space. Levine, Popovic, and Koltun [177] performs well in discrete-state MDPs, but is not suitable for continuous state spaces, does not operate on raw states but rather a hand-engineered set of integer component features, and requires the reward structure to be expressible as logical conjunctions. Meanwhile, Ratliff et al. [235] allows for larger state spaces and arbitrary continuous rewards, but still relies on engineering a relevant set of base features and severely underperforms in the case of non-expert human input when compared to more recent IRL techniques [178, 291]. Because of these shortcomings, IRL researchers have opted recently for either completely hand-specifying the features or using deep IRL for extracting them automatically from the raw continuous state space with non-expert demonstrations [102, 96].

Rather than relying on demonstrations for everything, we propose to first learn complex non-linear features leveraging explicit human input about relevant aspects of the task (Sec. 3.3). Based on these features, a reward can be inferred with minimal input (Sec. 3.4). Our results show that adding structure in such a targeted way can enhance both the generalization of the learned reward and data-efficiency of the method.

3.2 Problem Formulation

We consider a robot R operating in the presence of a human H from whom it is trying to learn to perform a task, ultimately seeking to enable autonomous execution. In the most general setting, both H and R are able to affect the evolution of the continuous state $s \in \mathbb{R}^d$ (i.e. robot joint poses or object poses) over time through their respective continuous actions a_H and a_R via a dynamics function f:

$$s^{t+1} = f(s^t, a_H^t, a_R^t) \ , \tag{3.1}$$

with $a_H \in \mathcal{A}_H$ and $a_R \in \mathcal{A}_R$, and $\mathcal{A}_H$ and $\mathcal{A}_R$ compact sets. Thus, when executing a task, the robot follows a trajectory $\xi = [s^0, a_H^0, a_R^0, s^1, a_H^1, a_R^1, \dots, s^T, a_H^T, a_R^T]$.

We assume that the human has some consistent internal preference ordering between different trajectories ξ, which affects the actions a_H that they choose. In principle, these human preferences could be captured by a reward function $R^*(\xi)$. Unfortunately, the robot does not have access to R^*, so to learn how to perform the task it must attempt to infer it. Since R^* may encode arbitrary preference orderings deeming the inference problem intractable, we assume that the robot reasons over a parameterized approximation R_θ induced by parameters $\theta \in \Theta$. The robot's goal is, thus, to estimate the human's preferred θ from their actions a_H.

Even with this parameterization, the space of possible reward functions is infinite-dimensional. One way to represent it using a finite θ is through the means of a finite family of basis functions Φ_i, also known as *features* [214]: $R_\theta(\vec{\Phi}(\xi))$, where $\vec{\Phi}$ is the set of chosen features Φ_i. Consistent with classical utility theories [282], we may decompose trajectory features Φ_i into state features ϕ_i and approximate the trajectory's reward through a cumulative return over time:

$$R_\theta(\xi) = R_\theta(\vec{\Phi}(\xi)) = \sum_{(s,a_H,a_R)\in\xi} r_\theta\big(\vec{\phi}(s, a_H, a_R)\big) \ . \tag{3.2}$$

This restriction of the representation to a finite set of features $\vec{\phi}$ is essentially a truncation of the infinite collection of basis functions spanning the full reward function space. Thus, the features we choose to represent the reward dramatically impact the reward functions that can be learned altogether. Importantly, this observation holds regardless of the representation power that r_θ has (linear combination, neural network, etc). Motivated by recovering a reward function r_θ that captures the person's preferences as best as possible, we are, thus, interested in the question of how to choose the feature representation $\vec{\phi}$.

We assume the robot's representation is initially a (possibly empty) set of features $\vec{\phi}$. In Sec. 3.3, we propose a protocol via which the robot can learn a novel feature to add to its existing set by soliciting representation-specific human input. We then describe classic offline IRL and its adaptation to situations where the human is teaching the reward from scratch (Sec. 3.4.1); our framework enables them to teach one feature at a time before teaching the reward on top using task demonstrations. Lastly, we present the online variant, where the robot executes the task according to a reward function defined on an incomplete feature set and the human intervenes to correct it (Sec. 3.4.2); our method allows them to explicitly focus on teaching the missing feature(s) and adding them to the set before the reward is updated.

3.3 Approach: Feature Learning with Feature Traces

We first look at learning individual feature functions. We focus on state features (ignoring actions from the feature representation), which we define as arbitrary complex mappings $\phi(s) : \mathbb{R}^d \rightarrow \mathbb{R}^+$. As such, in regions of the state space where the feature is highly expressed, this function has high positive values, whereas for states where the feature is not expressed, ϕ is closer to zero.

One natural idea for learning this mapping is treating it as a regression problem and asking the human for regression labels $(s, \phi(s))$ directly. Unfortunately, to learn anything useful, the robot would need a very large set of labels from the person, which would be too effortful for them to provide. Even worse, humans are notoriously unreliable at quantifying their preferences with any degree of precision [48], so their labels might result in arbitrarily noisy regressions. Hence, we need a type of human input that balances being informative and not placing too much burden on the human.

3.3.1 Feature Traces

To teach a non-linear representation of ϕ with little data, we introduce *feature traces* $\tau = s_{0:n}$, a novel type of human input defined as a sequence of n states that are monotonically decreasing in feature value, i.e. $\phi(s_i) \geq \phi(s_j), \forall i < j$. This approach relaxes the need for accurate state labeling, while simultaneously providing a combinatorial amount of state comparisons (see Sec. 3.3.2 for details) from each trace τ.

When learning a feature, the robot can query the human for a set T of N traces. The person gives a trace τ by simply moving the system from any start state s_0 to an end state s_n, noisily ensuring monotonicity. Our method, thus, only requires an interface for communicating ordered feature values over states: kinesthetic teaching is useful for household or small industrial robots, while teleoperation and simulation interfaces may be better for larger robotic systems.

To illustrate how a human might offer feature traces in practice, let's turn to Fig. 3.1 (Left). Here, the person is teaching the robot to keep the mug away from the laptop (i.e. not above). The person starts a trace at s_0 by placing the end-effector directly above the object center, then leads the robot away from the laptop to s_n. Our method works best when the person tries to be informative, i.e. covers diverse areas of the space: the traces illustrated move radially in all directions and start at different heights. While for some features, like distance from an object, it is easy to be informative, for others, like slowing down near objects, it might be more difficult. We explore how easy it is for users to be informative in our study in Sec. 3.5.2, with encouraging findings, and discuss alleviating existing limitations in Sec. 3.8.

The power of feature traces lies in their inherent structure. Our algorithm, thus, makes certain assumptions to harvest this structure for learning. First, we assume that the feature values of states along the collected traces $\tau \in$ T are monotonically decreasing. Secondly, we assume that by default the human starts all traces in states s_0 with the highest feature value across the domain, then leads the system to states s_n with the lowest feature value. In some situations, this assumption might unnecessarily limit the kinds of feature traces the human can provide. For example, the person might want to start somewhere where the feature is only "half" expressed relative to the feature range of the domain. Because of this, we optionally allow the human to provide *relative values* $v_0, v_n \in [0, 1]$[1] to communicate that the traces start/end at values that are fractions of the feature range of the domain.

3.3.2 Learning a Feature Function

To allow for arbitrarily complex non-linear features, we approximate a feature by a neural network $\phi_\psi(s) : \mathbb{R}^d \to \mathbb{R}^+$. We incorporate the assumptions in the previous section by training ϕ_ψ as a discriminative function with respect to the state ordering in feature traces $\tau \in \mathrm{T}$, and also encouraging the starts s_0 and ends s_n across all traces to have the same high and low values, respectively. For ease of exposition, we present our feature learning technique without the relative values v_0 and v_n first, then later describe how to modify the algorithm to include them.

3.3.2.1 Monotonicity Along Feature Traces. First, due to the monotonicity assumption along any feature trace $\tau_k = (s_0^k, s_1^k, \dots, s_n^k)$, when training ϕ_ψ we want to encourage feature values to decrease monotonically along every trace, i.e. $\phi_\psi(s_i^k) \geq \phi_\psi(s_j^k), \forall j > i, k$. For this purpose, we convert the set of collected traces $\tau_k \in \mathrm{T}$ into a dataset of *ordered* tuples $(s_i^k, s_j^k) \in \mathcal{T}_{ord}$, where every first element appears earlier in the trace than the second element (hence its feature value should be higher). This results in $\binom{(n+1)}{2}$ tuples per trace, which we can use for training ϕ_ψ.

We train the discriminative function ϕ_ψ as a predictor for whether a state s has a higher feature value than another state s', which we represent as a softmax-normalized distribution:

$$P(\phi_\psi(s) > \phi_\psi(s')) = P(s \succ s') = \frac{e^{\phi_\psi(s)}}{e^{\phi_\psi(s)} + e^{\phi_\psi(s')}} \quad , \tag{3.3}$$

where we define the shorthand notation $s \succ s'$ for $\phi_\psi(s) > \phi_\psi(s')$. We choose ψ to minimize a negative log-likelihood loss $L_{ord}(\psi)$ operating on the ordered tuples dataset:

$$L_{ord}(\psi) = -\sum_{(s,s')\in\mathcal{T}_{ord}} \log(P(s \succ s')) = -\sum_{(s,s')\in\mathcal{T}_{ord}} \log \frac{e^{\phi_\psi(s)}}{e^{\phi_\psi(s)} + e^{\phi_\psi(s')}} \quad . \tag{3.4}$$

Intuitively, this loss spaces out the feature values ϕ_ψ such that they decrease monotonically along every trace; however, this alone does not constrain the traces to have the same start and end values, respectively.

[1]Since specifying decimal fractions is difficult, the person gives percentages between 0 and 100 instead.

3.3.2.2 Start/End Feature Value Equivalence. To encourage all traces to start and end in the same high and low feature values, we need an additional loss encoding $\phi_\psi(s_0^i) = \phi_\psi(s_0^j)$ and $\phi_\psi(s_n^i) = \phi_\psi(s_n^j)$ for all $\tau_i, \tau_j \in \mathrm{T}$. We thus convert the set of collected traces T into another dataset $\mathcal{T}_{equiv}$ of *equivalence* tuples $(s_0^i, s_0^j), (s_n^i, s_n^j)\ \forall\ \tau_i, \tau_j \in \mathrm{T},\ i \neq j,\ i > j$. This results in $2\binom{N}{2}$ tuples where the states of the tuple (s, s') should have the same feature value, i.e. $\phi_\psi(s) = \phi_\psi(s')$. We denote this relationship as $s \sim s'$ to simplify notation.

When training ϕ_ψ, the predictor should not be able to distinguish which state has a higher feature value, hence $P(\phi_\psi(s) > \phi_\psi(s')) = 0.5$. As such, we introduce a second loss function $L_{equiv}(\psi)$ that minimizes the negative log-likelihood of both s having a higher feature value than s' and s' having a higher feature value than s:

$$L_{equiv}(\psi) = -\sum_{(s,s')\in\mathcal{T}_{equiv}} \log(P(s \succ s')) + \log(P(s' \succ s)) = -\sum_{(s,s')\in\mathcal{T}_{equiv}} \log \frac{e^{\phi_\psi(s)+\phi_\psi(s')}}{(e^{\phi_\psi(s)} + e^{\phi_\psi(s')})^2} \ . \tag{3.5}$$

This loss ensures the state space around feature trace starts and ends have similar feature values, respectively. [2]

We now have a total dataset $\mathcal{T} = \mathcal{T}_{ord} \cup \mathcal{T}_{equiv}$ of $|\mathcal{T}| = \sum_{i=1}^{N} \binom{(n^i+1)}{2} + 2\binom{N}{2}$ tuples, which is already significantly large for a small set of feature traces. We can use it to optimize a loss $L(\psi)$ that combines the ordered and equivalence losses:

$$L(\psi) = L_{ord}(\psi) + \lambda L_{equiv}(\psi)\ , \tag{3.6}$$

where λ is a hyperparameter trading off the two loss functions.

Given the loss function in (3.6), we can use any automatic differentiation package to compute its gradients and update ψ via gradient descent. Note that L_{equiv} is akin to a binary cross-entropy loss with a target of 0.5, whereas L_{ord} is similar to a binary cross-entropy loss with a target of 1. This form of loss function has been shown to be effective for preference learning [71, 142]. The key differences here are that our loss is over feature functions not rewards, and that preferences are state orderings provided via feature traces not trajectory comparisons. Additionally, in practice we normalize the feature functions to make their subsequent reward weights reflect importance relative to one another. We present the full feature learning algorithm using feature traces in Alg. 1.

3.3.2.3 Incorporating Relative Values. So far, we have assumed that all feature traces have starts and ends of the same high and low feature value, respectively. The optional relative values v_0, v_n can relax this assumption to enable the human to provide richer traces and teach more complex feature functions, e.g. where no monotonic path from the

[2]One could choose other losses to ensure equivalence of start and end values such as a p-norm $||\phi_\psi(s) - \phi_\psi(s')||_p$. We experimented with $p = 2$ but it produced inferior results.

Algorithm 1: Feature Learning via Feature Traces

```
Input: $N$ number of queries, $K$ iterations.
for $i \leftarrow 1$ to $N$ do
    Query feature trace $\tau$ as in Sec. 3.3.1.
    $\mathrm{T} \leftarrow \mathrm{T} \cup \tau$.
end
Convert T to datasets $\mathcal{T}_{ord}$ and $\mathcal{T}_{equiv}$ as in Sec. 3.3.2.
Initialize $\phi_\psi$ randomly.
for iteration $k \leftarrow 1$ to $K$ do
    Sample tuples batch $\hat{\mathcal{T}}_{ord} \in \mathcal{T}_{ord}$.
    Sample tuples batch $\hat{\mathcal{T}}_{equiv} \in \mathcal{T}_{equiv}$.
    Estimate $L(\psi)$ using $\hat{\mathcal{T}}_{ord}$, $\hat{\mathcal{T}}_{equiv}$, and (3.6).
    Update parameter $\psi$ via gradient descent on $L(\psi)$.
end
return normalized $\phi_\psi$
```

highest to lowest feature value exists. By default, $v_0 = 1$ communicating that the trace starts at the highest feature value of the domain, and $v_n = 0$ signifying that the trace ends at the lowest feature value. By allowing v_0 and v_n to be something different from their defaults, the person can provide traces that start at higher feature values or end at lower ones. We describe how to include these relative values in the feature training procedure in Sec. 3.9.1.1.

3.4 Approach: Reward Learning with Learned Features

Now that we have a method for learning features, we discuss how the robot can include this capability in reward learning frameworks. For exposition, we chose two reward learning frameworks – learning from demonstrations (offline) and from corrections (online) – but we stress that features learned with our method are applicable to any other reward learning method that admits features (e.g. comparisons, scalar feedback, etc.).

3.4.1 Offline FERL

We first consider the scenario where the human is attempting to teach the robot a reward function from scratch, i.e. the robot starts off with an empty feature set $\vec{\phi}$. For instance, imagine a system designer trying to engineer the robot's reward before deployment, or an end user resetting it and custom designing the reward for their home. We can think of this as an offline reward learning setting, where the person provides inputs to the robot before it starts executing the task. Here, we focus on learning from demonstrations, although our framework can be adapted to any other offline reward learning strategy.

In standard learning from demonstrations, deep IRL uses a set of demonstrations to train a reward function directly from the raw state, in an end-to-end fashion. Under our divide-and-conquer framework, we redistribute the human input the robot asks for: first ask for feature traces τ focusing explicitly on learning F features one by one via Alg. 1, and only then collect a few demonstrations $\xi \in \mathcal{D}^*$ to learn the reward on top of them. Alg. 2 summarizes the full procedure.

Algorithm 2: Offline FERL

```
Input: Demonstration set $\mathcal{D}^*$, $F$ number of features, $K$ iterations, $\alpha$ learning rate.
Initialize empty feature set $\vec{\phi} = []$.
for $f \leftarrow 1$ to $F$ do
    Learn feature $\phi_f$ using Alg. 1.
    $\vec{\phi} \leftarrow (\vec{\phi}, \phi_f)$.
end
Initialize $\theta$ randomly.
for iteration $k \leftarrow 1$ to $K$ do
    Generate samples $\mathcal{D}^\theta$ using current reward $R_\theta$.
    Estimate gradient $\nabla\mathcal{L}$ using $\mathcal{D}^*$, $\mathcal{D}^\theta$ in (3.13).
    Update parameter $\theta$ using gradient $\nabla\mathcal{L}$ in (3.14).
end
return optimized reward parameters $\theta$
```

3.4.1.1 Creating the Feature Set. Since the robot starts off with an empty feature set $\vec{\phi}$, the person has to teach it every relevant feature one at a time. To do so, they follow the procedure in Alg. 1, that is they collect a set of feature traces $\tau \in \mathrm{T}$ for the current feature, then use them to train ϕ_ψ. The person can add this new feature to the robot's existing set:

$$\vec{\phi} \leftarrow (\vec{\phi}, \phi_\psi) \ , \tag{3.7}$$

and repeat the procedure for as many features F as they want.

After being equipped with a new set of features taught by the human, the robot can undergo standard learning from demonstration procedures to recover the person's preferences. We now review Maximum Entropy IRL [311] for completion of the offline reward learning exposition.

3.4.1.2 Offline Reward Learning. To teach the robot the desired reward function R_θ, the person collects a set of demonstrations $\xi \in \mathcal{D}^*$ for how to perform the task by directly controlling the state s through their input a_H. During a demonstration, the robot is put in gravity compensation mode or teleoperated, to allow the person full control over the desired trajectory. The robot interprets the set of demonstrations $\mathcal{D}^*$ as evidence about the human's preferred θ parameter, and uses them to estimate it and, thus, to learn the reward function.

In order to reason about the human's preferences, the robot needs to be equipped with a model $P(\xi \mid \theta)$ for how those preferences affect their choice of demonstrations. For example, if the human were assumed to act optimally, the model would place all the probability on the set of trajectories that perfectly optimize the reward R_θ. However, since humans are not perfect, we relax this assumption and model them as being *noisily-optimal*, choosing trajectories that are approximately aligned with their preferences. We follow the Boltzmann noisily-rational decision model:

$$P(\xi \mid \theta, \beta) = \frac{e^{\beta R_\theta(\xi)}}{\int_{\bar{\xi}} e^{\beta R_\theta(\bar{\xi})} d\bar{\xi}} \ , \tag{3.8}$$

where the human picks trajectories proportional to their exponentiated reward [27, 282]. Here, $\beta \in [0, \infty)$ controls how much the robot expects to observe human input consistent with its reward model. For now, we use the Maximum Entropy IRL [311] version of this observation model where β is fixed to 1, so for notation simplicity we refer to this model as $P(\xi \mid \theta)$. Later in Sec. 3.4.2, we will allow β to vary and make use of it in the online version of our framework.

In maximum entropy IRL, to recover the θ parameter we maximize the log-likelihood $\mathcal{L}(\theta)$ of the observed data under the above model [146]. To see how, let's start by writing down the log-likelihood formula:

$$\mathcal{L}(\theta) = \log \prod_{\xi \in \mathcal{D}^*} P(\xi \mid \theta) = \sum_{\xi \in \mathcal{D}^*} \log \frac{e^{R_\theta(\xi)}}{\int_{\bar{\xi}} e^{R_\theta(\bar{\xi})} d\bar{\xi}} = \sum_{\xi \in \mathcal{D}^*} R_\theta(\xi) - |\mathcal{D}^*| \log \int_{\bar{\xi}} e^{R_\theta(\bar{\xi})} d\bar{\xi} \ . \tag{3.9}$$

Computing the integral over trajectories is intractable in real-world problems, so sample-based approaches to maximum entropy IRL estimate it with samples $\xi \in \mathcal{D}'$ drawn from a background distribution $q(\xi)$:

$$\mathcal{L}(\theta) \approx \sum_{\xi \in \mathcal{D}^*} R_\theta(\xi) - |\mathcal{D}^*| \log \frac{1}{|\mathcal{D}'|} \sum_{\bar{\xi} \in \mathcal{D}'} \frac{e^{R_\theta(\bar{\xi})}}{q(\bar{\xi})} \ . \tag{3.10}$$

The distribution $q(\xi)$ is chosen often times to be uniform; instead, we follow Finn, Levine, and Abbeel [96] and generate samples in those regions of the trajectory space that are good according to the current estimate of the reward function, i.e. $q(\xi) \propto e^{R_\theta(\xi)}$. We denote $\xi \in \mathcal{D}^\theta$ such a set sampled under θ.

We may now find θ by maximizing the log-likelihood $\mathcal{L}(\theta)$ using gradient-based optimization on the above objective. The gradient then takes the following form:

$$\nabla \mathcal{L} = \frac{1}{|\mathcal{D}^*|} \sum_{\xi \in \mathcal{D}^*} \nabla R_\theta(\xi) - \frac{1}{|\mathcal{D}^\theta|} \sum_{\bar{\xi} \in \mathcal{D}^\theta} \nabla R_\theta(\bar{\xi}) \ . \tag{3.11}$$

At this point, a standard deep IRL baseline could use any automatic differentiation package to compute the gradient and update the reward parameters directly from the raw trajectory state. Instead, consistent with prior work on reward learning with feature sets, we represent the reward as a linear combination of the learned features $\vec{\phi}$:

$$R_\theta(\xi) = \theta^T \vec{\Phi}(\xi) = \sum_{(s,a_H,a_R)\in\xi} \theta^T \vec{\phi}(s) \ . \tag{3.12}$$

Note that the linear reward assumption is not necessary for our algorithm to work. While in theory the reward could be modeled as non-linear, our divide-and-conquer approach is motivated by keeping the reward parameter space small while still effectively capturing the person's preferences.

For the linear case, the gradient becomes the difference between the observed demonstration feature values and the expected feature values dictated by the sampled trajectories:

$$\nabla\mathcal{L} = \frac{1}{|\mathcal{D}^*|}\sum_{\xi\in\mathcal{D}^*} \vec{\Phi}(\xi) - \frac{1}{|\mathcal{D}^\theta|}\sum_{\bar{\xi}\in\mathcal{D}^\theta} \vec{\Phi}(\bar{\xi}) \ . \tag{3.13}$$

Lastly, we compute an estimate $\hat{\theta}$ by iteratively computing the gradient $\nabla\mathcal{L}$ and updating the parameters until convergence:

$$\hat{\theta}' = \hat{\theta} - \alpha\left(\frac{1}{|\mathcal{D}^*|}\sum_{\xi\in\mathcal{D}^*} \vec{\Phi}(\xi) - \frac{1}{|\mathcal{D}^\theta|}\sum_{\bar{\xi}\in\mathcal{D}^\theta} \vec{\Phi}(\bar{\xi})\right) \ , \tag{3.14}$$

where α is the chosen learning rate. The final reward learning procedure, thus, consists of K iterations of generating samples $\mathcal{D}^\theta$ under the current reward, using them to estimate the gradient in (3.13), and updating the parameter θ via gradient descent with (3.14).

3.4.2 Online FERL

In Sec. 3.4.1, we saw that our method allows the person to specify a reward by sequentially teaching features and adding them to the robot's feature set before using demonstrations to combine them. However, in many situations the system designer or even the user teaching the features might not consider all aspects relevant for the task *a priori*. As such, we now consider an online reward learning version of our previous scenario, where the person provides inputs to the robot during the task execution and its feature space may or may not be able to correctly interpret them.

We assume the robot has access to an initial feature set $\vec{\phi}$, and is tracking a trajectory ξ optimizing its current estimate of the reward function R_θ in (3.12). If the robot is not executing the task according to the person's preferences, the human can intervene with input a_H. For instance, a_H might be an external torque that the person applies to change the robot's current configuration. Or, they might stop the robot and kinesthetically demonstrate the task, resulting in a trajectory. Building on prior work, we assume the robot can evaluate whether its existing feature space can explain the human input (Sec. 3.4.2.2). If it can, the robot directly updates its reward function parameters θ, also in line with prior work Bajcsy et al. [25] and Ratliff, Bagnell, and Zinkevich [237] (Sec. 3.4.2.1). If it can not, the human can teach the robot a new feature[3] ϕ_ψ just like in Sec. 3.4.1 and augment its feature set $\vec{\phi} \leftarrow (\vec{\phi}, \phi_\psi)$. The robot can then go back to the original human input a_H that previously could not be explained by the old features and use it to update its estimate of the reward parameters θ. Algorithm 3 summarizes the full procedure.

Algorithm 3: Online FERL

```
Input: Features $\vec{\phi} = [\phi_1, \dots, \phi_f]$, initial parameters $\theta$, confidence threshold $\epsilon$.
Plan initial trajectory $\xi$ by optimizing $R_\theta$.
while executing $\xi$ do
    if $a_H$ then
        Estimate confidence $\hat{\beta}$ from $a_H$ using (3.17).
        if $\hat{\beta} < \epsilon$ then
            Learn feature $\phi_{new}$ using Alg. 1.
            $\vec{\phi} \leftarrow (\vec{\phi}, \phi_{new}), \theta \leftarrow (\theta, 0.0)$.
        end
        Get induced trajectory $\xi_H$ from (3.15).
        Update parameter $\theta$ using $\xi_H$ in (3.16).
        Replan trajectory $\xi$ by optimizing new $R_\theta$.
    end
end
```

3.4.2.1 Online Reward Update. Whether it needs to learn a new feature ϕ_ψ or not, the robot has to then use the human input a_H to update its estimate of the reward parameters θ. Here, any prior work on online reward learning from user input is applicable, but we highlight one example to complete the exposition.

For instance, take the setting where the human's input a_H was an external torque, applied as the robot was tracking a trajectory ξ that was optimal under its current reward R_θ. Prior work Bajcsy et al. [25] has modeled this as inducing a deformed trajectory ξ_H, by propagating the change in configuration to the rest of the trajectory:

$$\xi_H = \xi + \mu A^{-1} \tilde{a}_H \ , \tag{3.15}$$

[3]Because feature learning was triggered by an intervention, it is fair to assume that the human knows what aspect of the task they were trying to correct.

where $\mu > 0$ scales the magnitude of the deformation, A defines a norm on the Hilbert space of trajectories[4] and dictates the deformation shape [88], and $\tilde{a}_H$ is a_H at the interaction time and 0 otherwise.

If we think of ξ_H as the human observation and of ξ as the expected behavior according to the current reward [25], we arrive at a natural alternation of the update rule in (3.14):

$$\hat{\theta}' = \hat{\theta} - \alpha\left(\vec{\Phi}(\xi_H) - \vec{\Phi}(\xi)\right) \ . \tag{3.16}$$

Intuitively, the robot updates its estimate $\hat{\theta}$ in the direction of the feature change induced by the human's correction a_H from ξ to ξ_H.

If instead, the human intervened with a full demonstration, work on online learning from demonstrations (Sec. 3.2 in Ratliff, Bagnell, and Zinkevich [237]) has derived the same update with ξ_H now being the human demonstration. In our implementation, we use corrections and follow Bajcsy et al. [24], which shows that people more easily correct one feature at a time, and only update the θ index corresponding to the feature that changes the most (after feature learning this is the newly learned feature). After the update, the robot replans its trajectory using the new reward.

3.4.2.2 Confidence Estimation. The robot can learn a new feature from the person because we assumed it has the capacity to detect that a feature is missing in the first place. We alluded earlier in Sec. 3.4.1 how this ability might be enabled by manipulating the β parameter in the observation model in (3.8). We now expand on this remark.

In the presented Boltzmann model, β controls how much the robot expects to observe human input consistent with its reward structure, and, thus, its feature space. A high β suggests that the input is consistent with the robot's feature space, whereas a low β may signal that no reward function captured by the feature space can explain the input. As such, inspired by work in Fridovich-Keil et al. [100], Fisac et al. [98], and Bobu et al. [38], instead of keeping β fixed like in the maximum entropy IRL observation model, we reinterpret it as a *confidence* in the robot's features' ability to explain human data.

When the human input a_H is a correction, following Sec. 7.4 (discussed later in Chapter 7), the robot estimates $\hat{\beta}$ by considering how efficient the human input a_H is in achieving the induced trajectory features $\vec{\Phi}(\xi_H)$. Accordingly, $\hat{\beta}$ is inversely proportional to the difference between the actual human input and the input that would have produced $\vec{\Phi}(\xi_H)$ optimally:

$$\hat{\beta} \propto \frac{1}{\|a_H\|^2 - \|a_H^*\|^2} \ , \tag{3.17}$$

where we obtain a_H^* by solving the optimization problem (7.21) in Chapter 7.

[4]We used a norm A based on acceleration, consistent with Bajcsy et al. [25], but other norm choices are possible as well.

Intuitively, if the person's input is close to the optimal a^*_H, then it achieves the induced features $\vec{\Phi}(\xi_H)$ efficiently, resulting in high confidence $\hat{\beta}$. If, however, there is a far more efficient alternative input – the difference between a_H and a^*_H is large –, $\hat{\beta}$ will be small: the person probably intended to give input about a feature the robot does not know about.

Alternatively, if the human input a_H is a demonstration, like in the classical IRL presented in Sec. 3.4.1, we may estimate $\hat{\beta}$ via a Bayesian belief update: $b'(\theta, \beta) \propto P(\xi \mid \theta, \beta) b(\theta, \beta)$. Once again, in our implementation we used corrections, but Chapter 7 shows confidence estimation can easily be adapted to learning from demonstrations if desired.

To detect a missing feature, the robot simply needs a confidence threshold ϵ. If $\hat{\beta} > \epsilon$, the robot is confident in its feature space, so it updates the reward as usual; if $\hat{\beta} < \epsilon$, its features are insufficient and the robot asks the person to be taught a new one.

3.5 Experiments: Learning Features with Feature Traces

Before testing FERL in the two reward learning settings of interest, we first analyze our method for learning features in experiments with a robotic manipulator. In Sec. 3.5.1, we inspect how well FERL can learn six different features of varying complexity by using real robot data collected from an expert – a person familiar with how the algorithm works. We then conduct an online user study in simulation in Sec. 3.5.2 to test whether non-experts – people not familiar with FERL but taught to use it – can teach the robot good features.

3.5.1 Expert Users

We have argued that feature traces are useful in teaching the robot features explicitly. In our first set of experiments, we look at how good the learned features are, and how their quality varies with the amount of feature traces provided.

3.5.1.1 Experimental Design. We conduct our experiments on a 7-DoF JACO robotic arm. We investigate six features in the context of personal robotics:

1. *table*: distance of the End-Effector (EE) to the table (T), as a z-coordinate difference: $EE^z - T^z$ (superscript denotes pose coordinate selection);

2. *coffee*: coffee cup upright orientation, defined by how far the EE is from pointing up: $1 - EE^R \cdot [0, 0, 1]$ (superscript denotes pose rotation matrix);

3. *laptop*: 0.3 meter xy-plane distance of the EE to a laptop (L), to avoid passing over the laptop: $\max\{0.3 - \|EE^{xy} - L^{xy}\|_2, 0\}$;

4. *test laptop location*: same as *laptop* but the test position differs from the training ones;

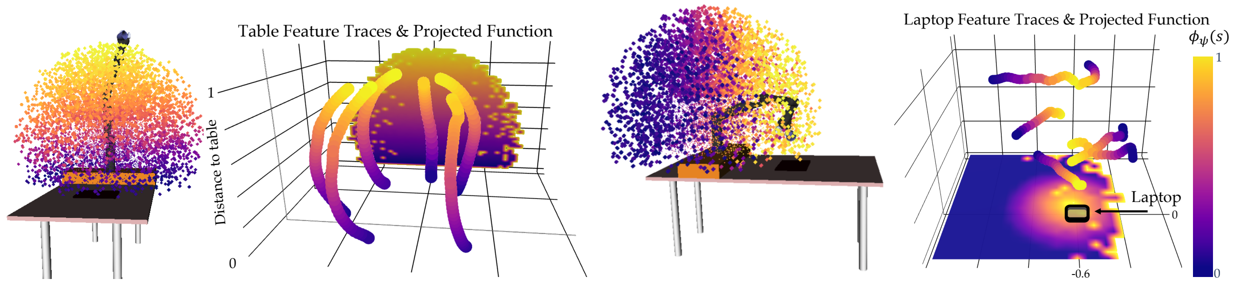


Figure 3.2: Visualization of the experimental setup, learned feature values $\phi_\psi(s)$, and training feature traces τ for *table* (up) and *laptop* (down). We display the feature values $\phi_\psi(s)$ for states s sampled from the reachable set of the 7-DoF arm, as well as their projections onto the yz and xy planes.

5. *proxemics*: non-symmetric 0.3 meter xy-plane distance between the EE and the human (H), to keep the EE away from them, three times as much when moving in front of them than on their side : $\max\{0.3 - \sqrt{\left(\frac{EE^y - H^y}{3}\right)^2 + (EE^x - H^x)^2}, 0\}$;
6. *between objects*: 0.2 meter xy-plane distance of the EE to two objects, O_1 and O_2 – the feature penalizes being above either object, and, to a lesser extent, passing in between the objects as defined by a distance to the imaginary line O_1O_2: $\max\{0.2 - \min\{0.8 * \|O_1O_2^{xy} - EE^{xy}\|_2, \|O_1^{xy} - EE^{xy}\|_2, \|O_2^{xy} - EE^{xy}\|_2\}, 0\}$.

Most features can be taught with the default relative values $v_n = 0$ and $v_0 = 1$, but *between objects* requires some traces with explicit values v_0, v_n. We approximate all features ϕ_ψ by neural networks (2 layers, 64 units each), and train them on a set of traces T using stochastic gradient descent (see Sec. 3.9.3.1 for training details details).

For each feature, we collected a set $\mathcal{F}$ of 20 feature traces (40 for the complex *test laptop location* and *between objects*) from which we sample subsets T $\in \mathcal{F}$ for training. We decide for each feature what an informative and intuitive set of traces would be, i.e. how to choose the starting states to cover enough of the space (details in Sec. 3.9.2.1). As described in Sec. 3.3.2, the human teacher starts at a state where the feature is highly expressed, e.g. for *laptop* that is the EE positioned above the laptop. They then move the EE away until the distance is equal to the desired radius. They do this for a few different directions and heights to give a diverse dataset.

Our raw state space consists of the 27D xyz positions of all robot joints and objects in the scene, as well as the rotation matrix of the EE. We assume known object positions but they could be obtained from a vision system. It was surprisingly difficult to train on both positions and orientations due to spurious correlations in the raw state space, hence we show results for training only on positions or only on orientations. This speaks to the need for methods that can handle correlated input spaces, which we expand on in Sec. 3.9.2.3.

Manipulated Variables. We are interested in seeing trends in how the quality of the learned features changes with more or less data available. Hence, we manipulate the

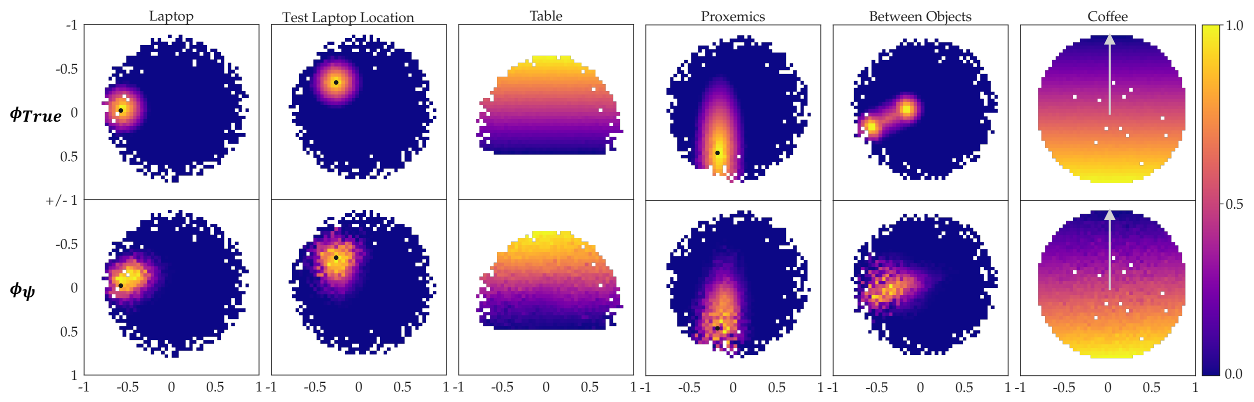


Figure 3.3: The plots display the ground truth ϕ_{True} (top rows) and learned feature values ϕ_ψ (bottom rows) over $\mathcal{S}_{\text{Test}}$, averaged and projected onto a representative 2D subspace: the xy-plane, the yz-plane (table), and the xz orientation plane for *coffee* (the arrow represents the cup upright).

number of traces N the learner gets access to.

Dependent Measures. After training a feature ϕ_ψ, we measure error compared to the ground truth feature ϕ_{True} that the expert tries to teach, on a test set of states $\mathcal{S}_{\text{Test}}$. To form $\mathcal{S}_{\text{Test}}$, we uniformly sample 10,000 states from the robot's reachable set. Importantly, many of these test points are far from the training traces, probing the generalization of the learned features ϕ_ψ. We measure error via the Mean Squared Error (MSE), MSE $= \frac{1}{|\mathcal{S}_{\text{Test}}|}\sum_{\mathcal{S}_{\text{Test}}} ||\phi_\psi(s) - \phi_{\text{True}}(s)||^2$. To ground the MSE values, we normalize them with the mean MSE of a randomly initialized untrained feature function, $\text{MSE}_{\text{norm}} = \frac{\text{MSE}}{\text{MSE}_{\text{random}}}$, hence a value of 1.0 is random performance. For each N, we run 10 experiments sampling different feature trace sets T from $\mathcal{F}$, and calculate MSE_{norm}.

Hypotheses.
H1: With enough data, FERL learns good features.
H2: FERL learns increasingly better features with more data.
H3: FERL becomes less input-sensitive with more data.

3.5.1.2 Qualitative Results. We first inspect the results qualitatively, for $N = 10$. In Fig. 3.2 we show the learned *table* and *laptop* features ϕ_ψ by visualizing the position of the EE for all 10,000 points in our test set. The color of the points encodes the learned feature values $\phi_\psi(s)$ from low (blue) to high (yellow): *table* is highest when the EE is farthest, while *laptop* peaks when the EE is above the laptop. In Fig. 3.3, we illustrate the Ground Truth (GT) feature values ϕ_{True} and the trained features ϕ_ψ by projecting the test points on 2D sub-spaces and plotting the average feature value per 2D grid point. For Euclidean features we used the EE's xy-plane or yz-plane (*table*), and for *coffee* we project the x-axis basis vector of the EE after forward kinematic rotations onto the xz-plane (arrow up represents the cup upright). White pixels are an artifact of sampling.

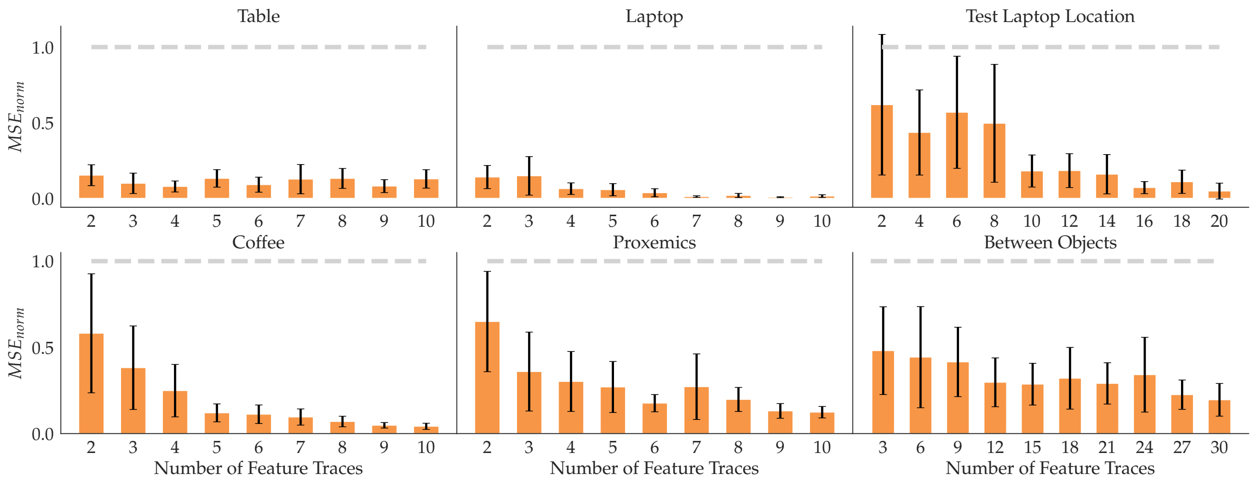


Figure 3.4: For each feature, we show the $\mathrm{MSE}_{\mathrm{norm}}$ mean and standard error across 10 random seeds with an increasing number of traces (orange) compared to random (gray).

We observe that ϕ_ψ resembles ϕ_{True} very well for most features. Our most complex feature, *between objects*, does not recreate the GT as well, although it does learn the general shape. However, we note in Sec. 3.9.4.1 that in smaller raw input space it is able to learn the fine-grained GT structure. This implies that spurious correlation in input space is a problem, hence for complex features more data or active learning methods to collect informative traces are required.

3.5.1.3 Quantitative Analysis. Fig. 3.4 displays the means and standard errors across 10 seeds for each feature with increasing amount of data N. To test H1, we look at the errors with the maximum amount of data. Indeed, FERL achieves small errors, put in context by the comparison with the error a random feature incurs (gray line). This is confirmed by an ANOVA with random vs. FERL as a factor and the feature ID as a covariate, finding a significant main effect ($F(1, 113) = 372.0123, p < .0001$). In line with H2, most features have decreasing error with increasing data. Indeed, an ANOVA with N as a factor and feature ID as a covariate found a significant main effect ($F(8, 526) = 21.1407, p < .0001$). Lastly, supporting H3, we see that the standard error on the mean decreases when FERL gets more data. To test this, we ran an ANOVA with the standard error as the dependent measure and N as a factor, finding a significant main effect ($F(8, 45) = 3.098, p = .0072$).

3.5.1.4 Summary. The qualitative and quantitative results support our hypotheses and suggest that our method requires few traces to reliably learn features ϕ_ψ that generalize well to states not seen during training. We also find that the more complex a feature, the more traces are needed for good performance: while *table* and *laptop* perform well with just $N = 4$, some other features, like *between objects*, require more traces. Active learning approaches that disentangle the learned function by querying traces at parts of the state space that are confusing could further reduce the amount of data required.

3.5.2 User Study

In the previous section, we have demonstrated that experts can teach the robot good feature functions. We now design a user study to test how well non-expert users can teach features with FERL and how easily they can use the FERL protocol.

3.5.2.1 Experimental Design. Due to COVID, we replicated our set-up from Fig. 3.1 (Left) in a pybullet simulator [74] in which users can move a 7 DoF-JACO robotic arm using their cursor. Through the interface in Fig. 3.5, the users can drag the robot to provide feature traces, and use the buttons for recording, saving, and discarding them.

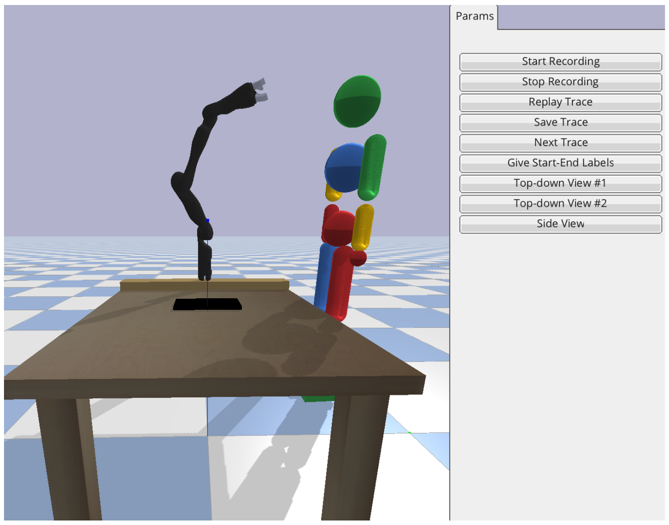


Figure 3.5: The pybullet simulator interface used in the user study, replicating our lab setup with the JACO robot.

The user study is split into two phases: familiarization and teaching. In the first phase, we introduce the user to the task context, the simulation interface, and how to provide feature traces through an instruction video and a manual. Next, we describe and 3D visualize the familiarization task feature *human* (0.3 meter xy-plane distance of the EE to the human position), after which we ask them to provide 10 feature traces to teach it. Lastly, we give the users a chance to see what they did well and learn from their mistakes by showing them a 3D visualization of their traces and the learned feature. See Sec. 3.9.2.4 for more details on the user training.

In the second phase, we ask users to teach the robot three features from Sec. 3.5.1: *table*, *laptop*, and *proxemics*. This time, we don't show the learned features until after all three tasks are complete.

Manipulated Variables. We manipulate the *input type* with three levels: *Random*, *Expert*, and *User*. For *Random*, we randomly initialize 12 feature functions per task; for *Expert*, the authors collected 20 traces per task in the simulator, then *randomly* subsampled 12 sets of 10 that lead to features of similar MSEs to the ones in the physical setup before; for *User*, each person provided 10 traces per task.

Dependent Measures. Our objective metric is the learned feature's MSE compared to the GT feature on $\mathcal{S}_{\text{Test}}$, similar to Sec. 3.5.1. Additionally, to assess the users' interaction

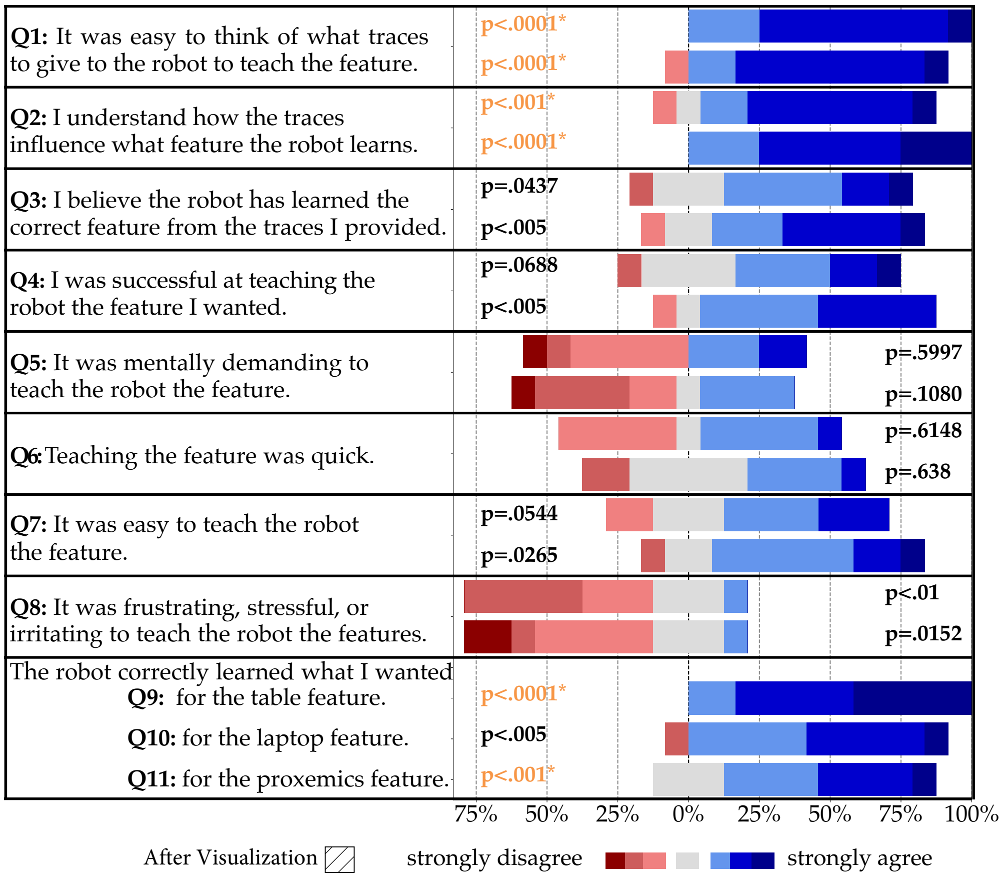


Figure 3.6: Questions, answer distributions, and p-values (2-sided t-test against the middle score 4) from the user study. The p-values in orange are significant after adjusted for multiple comparisons using the Bonferroni correction.

experience we administered the subjective 7-point Likert scale survey from Fig. 3.6, with some items inspired by NASA-TLX [126]. After they provide the feature traces for all 3 tasks, we ask the top eight questions in Fig. 3.6. The participants then see the 3D visualizations of their feature traces and learned features, and we survey all 11 questions as in Fig. 3.6 to see if their assessment changed.

Participants. We recruited 12 users (11 male, aged 18-30) from the campus community to interact with our simulated JACO robot and provide feature traces for the three tasks. All users had technical background, so we caution that our results will speak to FERL's usability with this population rather than the general population.

Hypotheses.
H4: FERL learns good features from non-expert user data.
H5: Users find it easy to think of traces to give the robot, believe they understand how these traces influence the learned feature, believe they were successful teachers, and find

our teaching protocol intuitive (little mental/physical effort, time, or stress).

3.5.2.2 Analysis.

Objective. Fig. 3.7 summarizes the results by showing how the MSE varies with each of our input types, for each task feature. Right off the bat, we notice that in line with H4, the MSEs for the user features are much closer to the expert level than to random. We ran an ANOVA with input type as a factor and task as a covariate, finding a significant main effect (F(2, 103) = 132.7505, p < .0001). We then ran a Tukey HSD post-hoc, which showed that the MSE for *Random* input was significantly higher than both *Expert* (p < .0001) and *User* (p < .0001), and found no significant difference between *Expert* and *User* (p = .0964). While this does not mean that user features are as good as expert features (we expect some degradation in performance when going to non-experts), it shows that they are substantially closer to them than to random, i.e. the user features maintain a lot of signal despite this degradation.

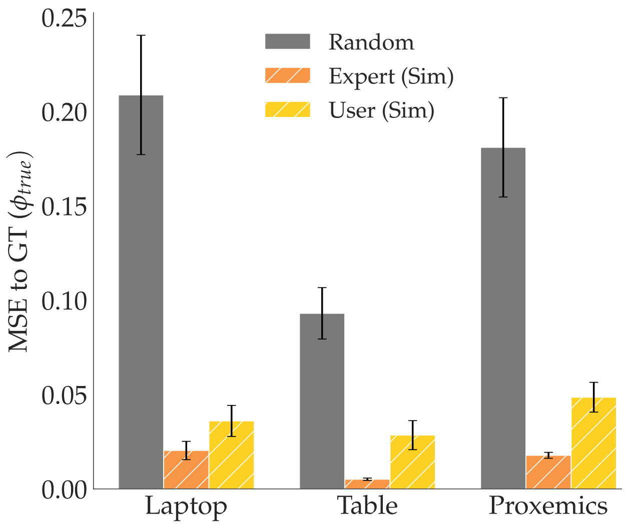


Figure 3.7: MSE to GT for the three features learned from expert (orange) and user (yellow) traces provided in simulation, and randomly (gray) initialized feature for comparison.

Subjective. In Fig. 3.6, we see the Likert survey scores before and after the users saw the teaching results. For every question, we report 2-sided t-tests against the neutral score 4. These results support H5, although the evidence for finding the teaching protocol intuitive is weaker, and participants might have a bias to be positive given they are in a study. In fact, several participants mentioned in their additional remarks that they had a good idea of what traces to give, and the only frustrating part was the GUI interface, which was necessary because in-person studies are not possible during the COVID pandemic ("I had a pretty good mental model for what I wanted to show, but found it frustrating doing that with a mouse", "I know what it wants, but the interface makes it difficult to give those exact traces"); performing the experiment as it was originally intended with the real robot arm would have potentially alleviated this issue ("With manual control of the arm it would have been a lot easier.").

Looking before and after the visualization, we find a trend: seeing the result seems to reinforce people's belief that they were effective teachers (Q3, Q4), also noticed in their comments ("Surprising how well it learned!", "Surprised that with limited coverage it generalized pretty well."). Also, in support of H4, we see significant evidence that users thought the robot learned the correct feature (Q9-Q11).

Lastly, we wanted to know if there was a correlation between subjective scores and objective performance. We isolated the "good teachers" – the participants who scored better than average on all 3 feature tasks in the objective metric, and compared their subjective scores to the rest of the teachers. By running a factorial likelihood-ratio test for each question, we found a significant main effect for good teachers: they are more certain that the robot has learned a correct feature even before seeing the results (Q3, p = .001), are more inclined to think they were successful (Q4, p = .0203), and find it significantly easier to teach features (Q7, p = .0202).

3.5.2.3 Summary. Both the objective and subjective results provide evidence that non-expert users can teach the robot reasonable features using our FERL protocol. In addition, participants found our teaching protocol intuitive, suggesting that feature traces can be useful for teaching features outside of the system designer's setting. In the following sections, we explore whether both expert and non-expert features can be used to improve reward learning generalization.

3.6 Experiments: Online FERL

Now that we have tested our method for learning features with both experts and non-experts, we analyze how the learned features affect reward learning. In this section, we start with the easier setting where the robot already has a feature set that it is using for online reward learning, but the human might provide input about a missing feature.

3.6.1 Expert Users

When the robot receives human input that cannot be explained by its current set of features, we hypothesize that adding FERL features to it can induce structure in the reward learning procedure that helps better recover the person's preferences. We first test this hypothesis with expert user data.

3.6.1.1 Experimental Design. We run experiments on the same JACO robot arm in three settings in which two features are known ($\phi_{\text{coffee}}, \phi_{\text{known}}$) and one is unknown. In all tasks, the true reward is $r_{\text{true}} = (0, 10, 10)(\phi_{\text{coffee}}, \phi_{\text{known}}, \phi_{\text{unknown}})^T$. We include ϕ_{coffee} with zero weight to evaluate if the methods can learn to ignore irrelevant features. In task 1, ϕ_{laptop} is unknown and the known feature is ϕ_{table}; in task 2, ϕ_{table} is unknown and ϕ_{laptop} is known; and in task 3, $\phi_{\text{proxemics}}$ is unknown and ϕ_{table} is known. We name the tasks *Laptop Missing*, *Table Missing*, and *Proxemics Missing*, respectively.

Manipulated Variables. We manipulate the *learning method* with 2 levels: FERL and an adapted ME-IRL baseline[5] [96, 291] learning a deep reward function from demonstrations. We model the Maximum Entropy Inverse Reinforcement Learning (ME-IRL) reward function r_ω as a neural network with 2 layers, 128 units each. For a fair comparison, we gave r_ω access to the known features: once the 27D Euclidean input is mapped to a neuron, a last layer combines it with the known feature vector.

Also for a fair comparison, we collected a set of demonstrations for ME-IRL designed to be as informative as possible: we chose diverse start and goal configurations for the demonstrations, and focused some of them on the unknown feature and some on learning a combination between features (see Sec. 3.9.2.2). Moreover, FERL and ME-IRL rely on different input types: FERL on feature traces τ and pushes a_H and ME-IRL on a set of near-optimal demonstrations $\mathcal{D}^*$. To level the amount of data each method has access to, we collected the traces T and demonstrations $\mathcal{D}^*$ such that ME-IRL has more data points: the average number of states per demonstration/trace were 61 and 31, respectively.

Following (3.11), the gradient of the ME-IRL objective with respect to the reward parameters ω can be estimated by: $\nabla_\omega \mathcal{L} \approx \frac{1}{|\mathcal{D}^*|}\sum_{\tau\in\mathcal{D}^*}\nabla_\omega R_\omega(\tau) - \frac{1}{|\mathcal{D}^\omega|}\sum_{\tau\in\mathcal{D}^\omega}\nabla_\omega R_\omega(\tau)$ [291, 96]. Here, $R_\omega(\tau)=\sum_{s\in\tau} r_\omega(s)$ is the parametrized reward, $\mathcal{D}^*$ the expert demonstrations, and $\mathcal{D}^\omega$ are trajectory samples from the r_ω induced near optimal policy. We use TrajOpt [255] to obtain the samples $\mathcal{D}^\omega$ (see Sec. 3.9.3.4 for details). For practical considerations and implementation details of the online version of FERL we used, see Sec. 3.9.3.2.

Dependent Measures. We compare the two reward learning methods across three metrics commonly used in the IRL literature [69]: 1) *Reward Accuracy*: how close to GT the learned reward is, 2) *Behavior Accuracy*: how well do the behaviors induced by the learned rewards compare to the GT optimal behavior, measured by evaluating the induced trajectories on GT reward, and 3) *Test Probability*: how likely trajectories generated by the GT reward are under the learned reward models.

For *Reward Accuracy*, note that any affine transformation of a reward function would result in the same induced behaviors, so simply measuring the MSE between the learner's reward and the GT reward may not be informative. As such, we make reward functions given by different methods comparable by computing each learner's reward values on $\mathcal{S}_{\text{Test}}$ and normalizing the resulting set of rewards to be in $[0,1]$. This allows us to compute the MSE on $\mathcal{S}_{\text{Test}}$ between each method and the GT. Similarly to Sec. 3.5.1, we report this metric by varying the number of traces / demonstrations each learner gets access to. For *Behavior Accuracy* and *Test Probability*, we train FERL and ME-IRL with a set of 10 traces / demonstrations. For *Behavior Accuracy*, we use TrajOpt [255] to produce optimal trajectories for 100 randomly selected start-goal pairs under the learned rewards. We evaluate the trajectories with the GT reward r_{true} and divide by the reward of the GT induced trajectory

[5]We chose ME-IRL as it is the state-of-the-art method for learning rewards and does not rely on base feature engineering, as explained in Section 3.1. We also tried a linear variant of ME-IRL optimizing the reward parameters on top of random features modeled as neural networks. However, we found the performance of this alternate baseline to be consistently inferior to that of the deep ME-IRL (see Sec. 3.9.4.2), so we only compare against the deep variant.

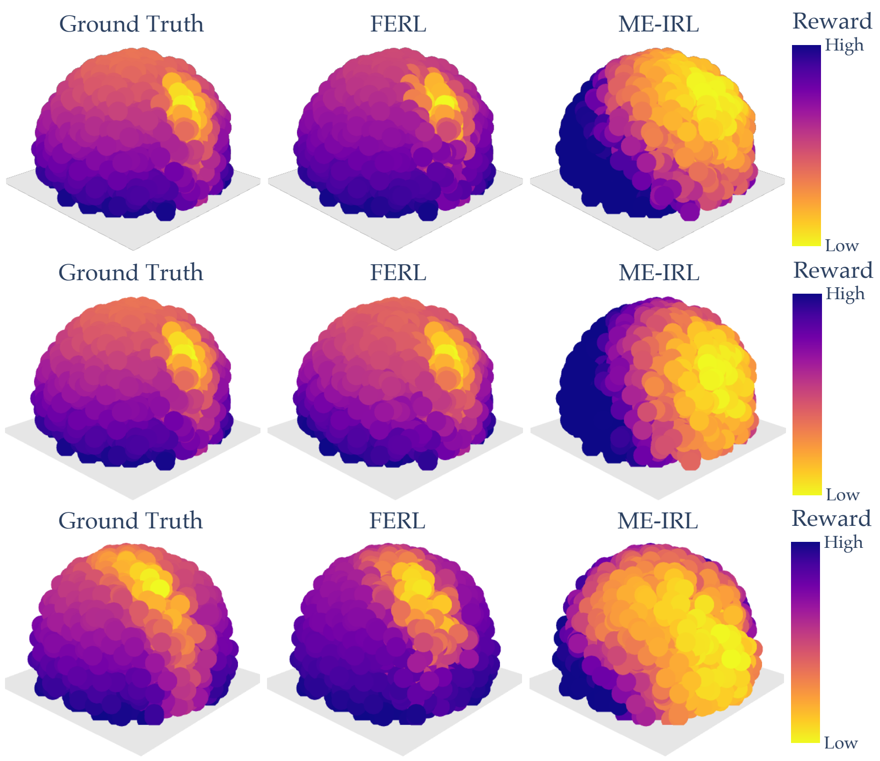


Figure 3.8: Visual comparison of the ground truth, online FERL, and ME-IRL rewards for *Laptop Missing* (top), *Table Missing* (middle) and *Proxemics Missing* (bottom).

for easy relative comparison. For *Test Probability*, we generate 100 optimal trajectories using the GT reward, then evaluate their likelihood under the Boltzmann model in (3.8) with each learned reward. To approximate the intractable integral in (3.8), we sample[6] sets of 100 trajectories for every start-goal pair corresponding to the optimal trajectories. For a fair comparison, we use the normalized rewards once again, and fit the maximum likelihood coefficient $\hat{\beta}$ for each model.

Hypotheses.
H6: Online FERL learns rewards that better generalize to the state space than ME-IRL.
H7: Online FERL performance is less input-sensitive than ME-IRL's.

3.6.1.2 Qualitative Comparison. In Fig. 3.8, we show the learned FERL and ME-IRL rewards as well as the GT for all three tasks evaluated at the test points. As we can see, by first learning the missing feature and then the reward on the extended feature

[6]To obtain dynamically feasible trajectories, we sampled random objectives given by linear combinations of various features, and optimized them with TrajOpt. While this sampling strategy cannot be justified theoretically, it works well in practice: the resulting optimized trajectories are a heuristic for sampling diverse and interesting trajectories in the environment.

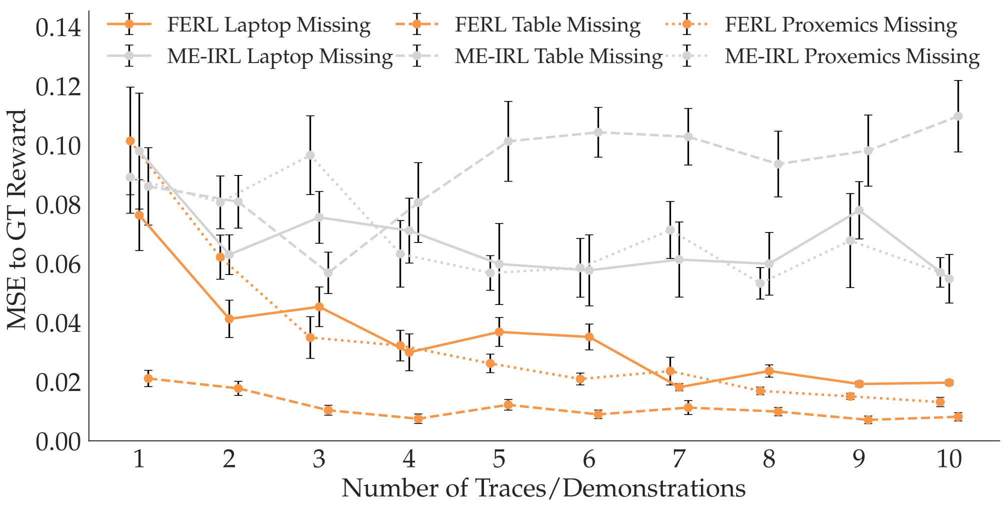


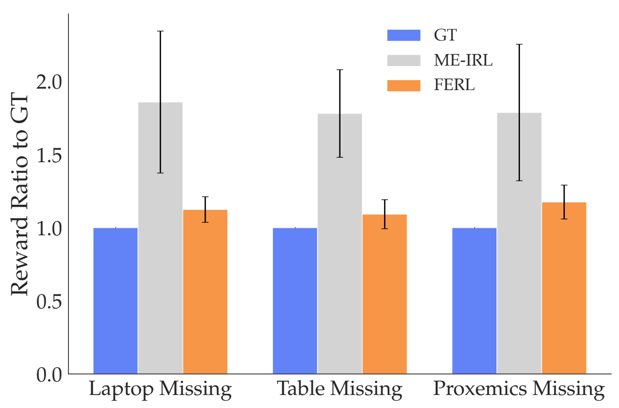
Figure 3.9: MSE of online FERL and ME-IRL to GT reward across all three tasks. FERL learns rewards that better generalize to the state space.

vector, FERL is able to learn a fine-grained reward structure closely resembling the GT. Meanwhile, ME-IRL learns some structure capturing where the laptop or the human is, but not enough to result in a good trade-off between the active features.

3.6.1.3 Quantitative Analysis. To compare *Reward Accuracy*, we show in Fig. 3.9 the MSE mean and standard error across 10 seeds, with increasing training data. We visualize results from all 3 tasks, with FERL in orange and ME-IRL in gray. FERL is closer to GT than ME-IRL no matter the amount of data, supporting H6. To test this, we ran an ANOVA with learning method as the factor, and with the task and data amount as covariates, and found a significant main effect (F(1, 595) = 335.5253, p < .0001).

Additionally, the consistently decreasing MSE in Fig. 3.9 for FERL suggests that our method gets better with more data; in contrast, the same trend is inexistent with ME-IRL. Supporting H7, the high standard error that ME-IRL displays implies that it is highly sensitive to the demonstrations provided and the learned reward likely overfits to the expert demonstrations. We ran an ANOVA with standard error as the dependent measure, focusing on the $N = 10$ trials which provide the maximum data to each method, with the learning method as the factor and the task as a covariate. We found that the learning method has a significant effect on the standard error (F(1, 4) = 12.1027, p = .0254). With even more data, this shortcoming of IRL might disappear; however, this would pose an additional burden on the human, which our method successfully alleviates.

We also looked at *Behavior Accuracy* for the two methods. Fig. 3.10 (left) illustrates the reward ratios to GT for all three tasks. The GT ratio is 1 by default, and the closer to 1 the ratios are, the better the performance because all rewards are negative. The figure further supports H6, showing that FERL rewards produce trajectories that are preferred under



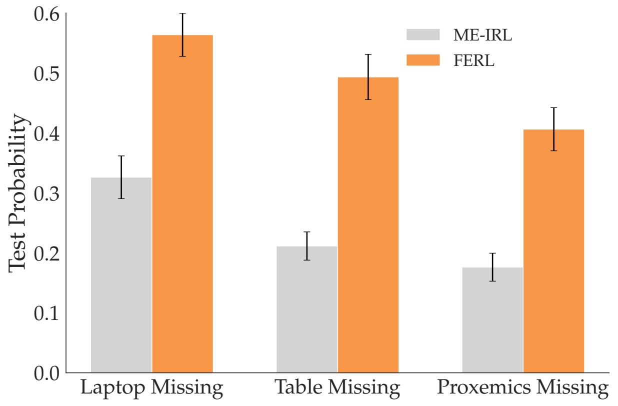


Figure 3.10: (Left) Induced trajectories' reward ratio for the two methods compared to GT. ME-IRL struggles to generalize across all tasks. (Right) Probability assigned by the two methods to a set of optimal trajectories under the Boltzmann assumption. The trajectories are more likely under FERL than ME-IRL, suggesting FERL is the more accurate reward model.

the GT reward over ME-IRL reward trajectories. An ANOVA using the task as a covariate reveals a significant main effect for the learning method (F(1, 596) = 14.9816, p = .0001).

Lastly, we compare how likely a test set of trajectories given by optimizing the GT reward is under the two models. A more accurate reward model should give higher probabilities to the demonstrated trajectories under the Boltzmann noisily-rational assumption in (3.8). Fig. 3.10 (right) illustrates that FERL does indeed assign higher likelihood to the test trajectories than ME-IRL, which is consistent with H6.

3.6.1.4 Summary. The rewards learned with FERL qualitatively capture more structure than ME-IRL ones, but they also quantitatively get closer to the GT. Using FERL features – at least when the robot is missing one feature – seems to induce useful structure in the reward learning process that guides the robot to better capture the person's preferences. These results hold when the person teaching the missing feature is an expert user; we next look at the case where a novice interacts with the robot instead.

3.6.2 Non-expert Users

The objective results in Sec. 3.5.2 show that while users' performance degrades from expert performance, they are still able to teach features with a lot of signal. We now want to test how important the user-expert feature quality gap is when it comes to using these features for online reward learning.

3.6.2.1 Experimental Design. For this experiment, we had a similar setup to the one in Sec. 3.6.1, only that we performed reward learning with FERL using the user-taught simulation features from the user study. We wanted to see if the divide-and-conquer approach employed by FERL results in better rewards than ME-IRL even when using noisy simulation data.

Manipulated Variables. We manipulate the *learning method*, FERL or ME-IRL, just like in Sec. 3.6.1. Because corrections and demonstrations would be very difficult in simulation, we use for ME-IRL the expert data from the physical robot. For FERL, we use the user data from the simulation, and the expert corrections that teach the robot how to combine the learned feature with the known ones. Note that this gives ME-IRL an advantage, since its data is both generated by an expert, and on the physical robot. Nonetheless, we hypothesize that the advantage of the divide-and-conquer approach is stronger.

Dependent Measures. We use the same objective metric as *Reward Accuracy* in the expert comparison in Sec. 3.6.1: the learned reward MSE to the GT reward on $\mathcal{S}_{\text{Test}}$.

Hypothesis.
H8: Online FERL learns more generalizable rewards than ME-IRL even when using features learned from data provided by non-experts in simulation.

3.6.2.2 Analysis. Fig. 3.11 illustrates our findings for the reward comparison. In the figure, we added FERL with expert-taught simulation features for reference: we randomly subsampled sets of 10 from 20 expert traces collected by the authors, and trained 12 expert features for each of our 3 task features. We see that, *even though ME-IRL was given the advantage of using physical expert demonstrations, it still severely underperforms when compared to FERL with both expert and user features learned in simulation.* This finding is crucial because it underlines the power of our divide-and-conquer approach in online reward learning: even when given imperfect features, the learned reward is superior to trying to learn everything implicitly from demonstrations.

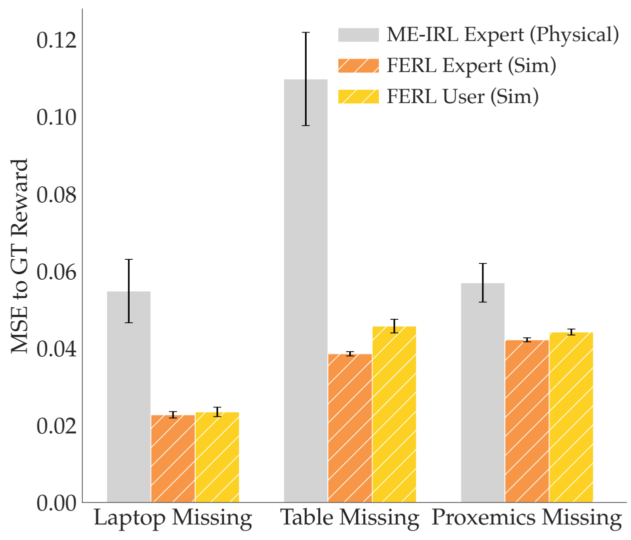


Figure 3.11: MSE to GT reward for the three tasks, comparing ME-IRL from expert physical demonstrations (gray) to online FERL from expert (orange) and non-expert (yellow) features learned in simulation and combined via corrections.

We verified this result with an ANOVA with the learning method as factor and the task as covariate. We found a significant main effect for the learning method ($F(1, 62) = 41.2477$, $p < .0001$), supporting H8.

3.6.2.3 Summary. Despite the degradation in feature quality we see in user features when compared to expert ones, we find that the structure they do maintain is advantageous in online reward learning. This suggests that the online instantiation of FERL can be used even by non-experts to better teach the robot their preferences.

3.7 Experiments: Offline FERL

In the online reward learning setting, the robot was already equipped with a starting feature set, and we tested how learning missing features affects the reward. We now look at the scenario where the robot's reward must be programmed entirely from scratch, teaching each feature separately before combining them into a reward via demonstrations.

3.7.1 Expert Users

We have argued that learned features can induce useful structure that speeds up reward learning. We test how the reward is affected when the entire structure is built up from the expert features taught from real robot data in Sec. 3.5.1.

3.7.1.1 Experimental Design. We run experiments on the robot arm in three settings of increasing complexity: in the first, the true reward depends on a single feature, and every subsequent task adds another feature to the reward. In task 1, the true reward depends on only ϕ_{table}. In task 2, we add the ϕ_{laptop} feature, and in task 3 the $\phi_{\text{proxemics}}$ feature. In both tasks 2 and 3, the reward equally combines the two and three features, respectively. Task 1 should be easy enough for even an end-to-end IRL method to solve, especially since it relies on the simplest feature that we have considered. Meanwhile, tasks 2 and 3 require learning rewards that are more structurally complex. We name the three tasks *One Feature*, *Two Features*, and *Three Features*, respectively.

Manipulated Variables. We manipulated the *learning method* with 2 levels: FERL and ME-IRL. While in Sec. 3.6 ME-IRL had access to the known features, this time the reward network is a function mapping directly from the 27D Euclidean input space only. For practical considerations and implementation details of the offline version of FERL we used, see Sec. 3.9.3.3.

For a fair comparison, we once again took great care in how we collected the demonstrations ME-IRL learns from. Just like before, we chose diverse start and goal configurations, and focused some of the demonstrations on each individual feature, and, when it applies, on each combination of features (see Sec. 3.9.2.2). Importantly, while ME-IRL uses a set of near-optimal demonstrations $\mathcal{D}^*$, FERL requires both demonstrations and feature traces τ. To level the amount of data each method has access to, we distributed the demonstrations and traces FERL has access to such that ME-IRL has more data points. The average number of states per demonstration/trace were 64 and 31, respectively, so if we keep the number of ME-IRL demonstrations and FERL traces the same, FERL has a non-zero budget of demonstrations to use for cases with more than one demonstration ($N > 1$).

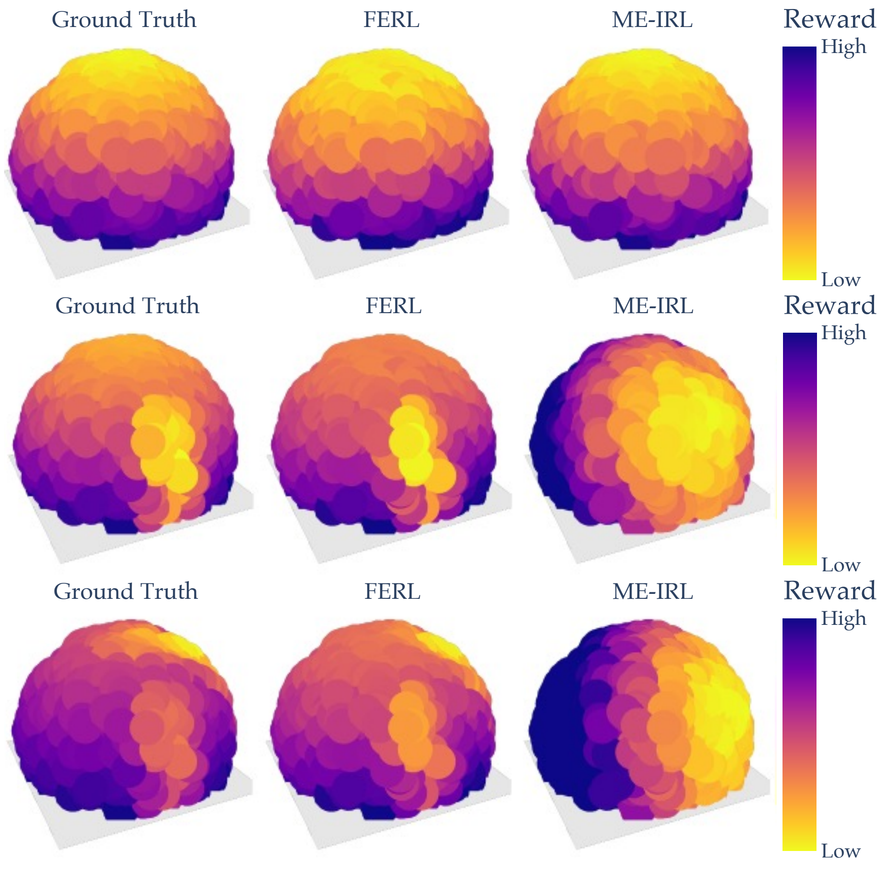


Figure 3.12: Visual comparison of the ground truth, offline FERL, and ME-IRL rewards for *One Feature* (top), *Two Features* (middle) and *Three Features* (bottom).

Dependent Measures. We use the same objective metrics as *Reward Accuracy*, *Behavior Accuracy*, and *Test Probability* in Sec. 3.6.1. For *Reward Accuracy*, we vary the number N of traces / demonstrations each learner gets but skip $N = 1$ because FERL would have an unfair advantage in the amount of data given. We give ME-IRL up to 10, 20, and 30 demonstrations for the three tasks, respectively. Meanwhile, we give FERL up to 10 traces for each feature, and 1, 2, and 3 demonstrations for each task, respectively. Overall, FERL would use up to 10 traces and one demonstration, up to 20 traces and 2 demonstrations, and up to 30 traces and 3 demonstrations, while ME-IRL would be given 10, 20, and 30 demonstrations for each task, respectively. For *Behavior Accuracy* and *Test Probability*, we train FERL with 10 traces per feature and 1, 2, or 3 demonstrations, and ME-IRL with 10, 20, and 30 demonstrations, respectively. Just like in Sec. 3.6.1, for *Behavior Accuracy* we produce optimal trajectories for 100 randomly selected start-goal pairs under the learned rewards and evaluate them under the GT reward. Meanwhile, for *Test Probability*, we generate 100 optimal trajectories using the GT reward, then evaluate their likelihood under the learned models.

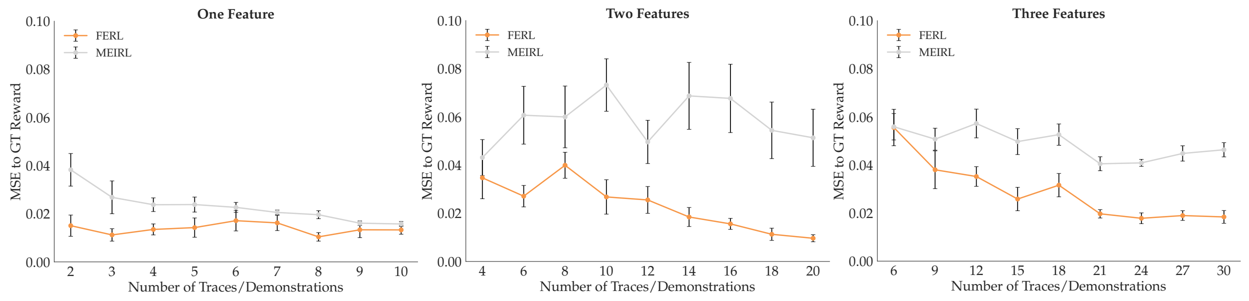


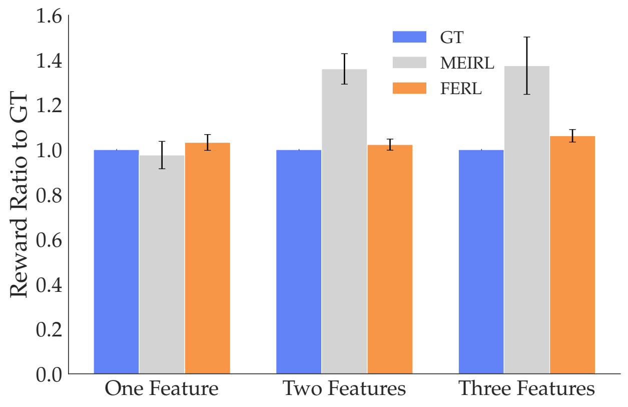
Figure 3.13: MSE of offline FERL and ME-IRL to GT reward for *One Feature* (Left), *Two Features* (Middle), and *Three Features* (Right). In most data regimes, FERL learns rewards that better generalize to the state space.

Hypothesis.
H9: Offline FERL learns rewards that better generalize to the state space than ME-IRL.

3.7.1.2 Qualitative Comparison. In Fig. 3.12, we show the learned FERL and ME-IRL rewards as well as the GT for all three tasks evaluated at the test points. The figure illustrates that by first learning each feature separately and then the reward that combines them, FERL is able to learn a fine-grained reward structure closely resembling the GT. For the easiest task, *One Feature*, ME-IRL does recover the GT appearance, but this is unsurprising since the *table* feature is very simple. For the other more complex two tasks, just like in the online case, ME-IRL learns some structure capturing where the laptop or the human is, but not enough to result in a good trade-off between the features.

3.7.1.3 Quantitative Analysis. To compare *Reward Accuracy*, we show in Fig. 3.13 the MSE mean and standard error across 10 seeds, with increasing training data. We visualize results from all 3 tasks side by side, with FERL in orange and ME-IRL in gray. For *One Feature*, as expected, ME-IRL does eventually learn a good reward with enough data. However, for the other more complex tasks that combine multiple features, ME-IRL underperforms when compared to our method. Overall, across the tasks, FERL is closer to GT than ME-IRL no matter the amount of data, supporting H9. To test this, we ran an ANOVA with learning method as the factor, and with the task and data amount as covariates, and found a significant main effect ($F(1, 535) = 148.8431$, $p < .0001$).

For comparing *Behavior Accuracy*, Fig. 3.14 (left) illustrates the reward ratios to GT for all three tasks. When the reward consists of a single very simple feature, ME-IRL performs just as well as our method. However, when the reward structure more complexly combines multiple features, ME-IRL does not produce as good trajectories under the GT reward as FERL, supporting H8. We ran an ANOVA using the learning method as a factor and the task as a covariate and did not find a significant main effect, probably due to the *One Feature* results. To verify this theory, we re-ran the ANOVA using only the data from the more complex *Two Features* and *Three Features* tasks, and did, in fact, find a significant main



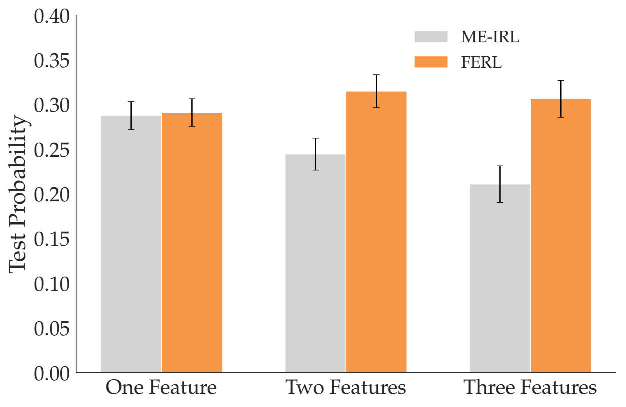


Figure 3.14: (Left) Induced trajectories' reward ratio for the two methods compared to GT. While ME-IRL generalizes for the single feature task, it struggles with the more complex multiple feature tasks. (Right) Probability assigned by the two methods to a set of optimal trajectories under the Boltzmann assumption. For the more complex multiple feature tasks, the trajectories are more likely under FERL than ME-IRL.

effect (F(1, 397) = 5.7489, p = .0097). Results with the *Test Probability* metric paint a similar picture. Fig. 3.14 (right) shows that for the easy *One Feature* case, both methods perform comparably, but when the reward is more complex (*Two Features* and *Three Features*), FERL outperforms ME-IRL and assigns higher probability to the test trajectories.

3.7.1.4 Summary. The results in this section suggest that while ME-IRL is capable of recovering very simple reward structures, it does not perform as well as using FERL features for complex rewards. This observation applies when the features are taught by experts, so we now test what happens if we instead use non-expert user features.

3.7.2 Non-expert Users

In Sec. 3.6.2, we saw that user-taught FERL features have enough structure to help the robot recover the human's preferences in online reward setting where the original feature set is incomplete. However, there we only had one missing feature. In this section, we test the more challenging scenario, where we learn a reward from scratch using the noisy user features learned in simulation.

3.7.2.1 Experimental Design. For this experiment, we had a similar setup as in Sec. 3.6.2 – using the user-taught simulation features for learning the reward – only this time we tested the offline instantiation of FERL. Given that now we combine multiple noisy features together into a reward, we wanted to see how our divide-and-conquer approach fares against the ME-IRL baseline.

Manipulated Variables. We manipulate the *learning method*, FERL or ME-IRL, just like in Sec. 3.7.1. Like in Sec. 3.6.2, we use demonstrations collected from the expert on the physical robot for ME-IRL. For FERL, we use the user data from the simulation, and the expert demonstrations that teach the robot how to combine the learned feature into a reward. Note that this gives ME-IRL an advantage, since all its data is both generated by an expert, and on the physical robot.

Dependent Measures. We use the same objective metric as *Reward Accuracy* in the expert comparison in Sec. 3.7.1: the learned reward MSE to the GT reward on $\mathcal{S}_{\text{Test}}$.

Hypotheses.
H10: Offline FERL learns more generalizable rewards than ME-IRL even when using features learned from data provided by non-experts in simulation.

3.7.2.2 Analysis. Fig. 3.15 illustrates our findings for the reward comparison. We also added the offline FERL reward using expert-taught simulation features for reference, where we randomly subsampled sets of 10 traces and trained 12 expert features for each of the three features. This time, we find that the user features are noisy enough that, when combined into a reward, they do not reliably provide an advantage over ME-IRL. This could be attributed to the difficulty of teaching features in a simulator, especially given that there is no easy way to approximate distances and traces in 3D space with a 2D interface are hard. We verified this result with an ANOVA with the learning method as a factor and the task as a covariate, and, as expected, we found no significant main effect.

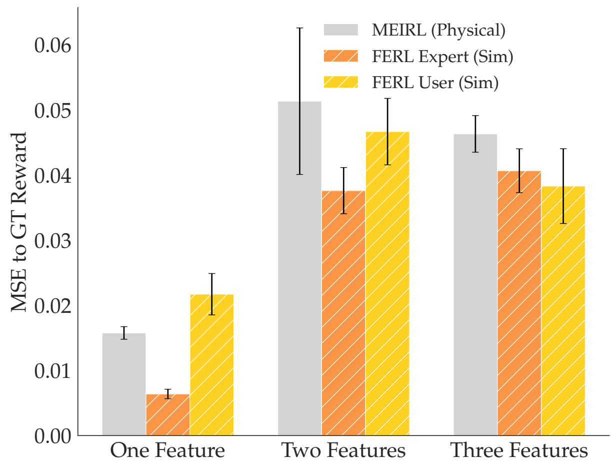


Figure 3.15: MSE to GT reward for the three tasks, comparing ME-IRL from expert physical demonstrations (gray) to offline FERL from expert (orange) and non-expert (yellow) features learned in simulation and combined via corrections.

3.7.2.3 Summary. Previously, we have seen how structure can indeed help reward learning generalizability and sample efficiency; but we now see that the *wrong* – or very noisy – structure obtained from traces from simulation may diminish the benefits that our divide-and-conquer approach promises. However, we suggest taking this result with a grain of salt, since ME-IRL had the advantage of all-expert, all-physical data, whereas our method was limited to data collected in simulation from novice users. While not possible during the pandemic, we are optimistic that with physical demonstrations the benefits would be more prominent.

3.8 Discussion

Learning reward functions is a popular way to help robots generate behaviors that adapt to new situations or human preferences. In this work, we propose that robots can learn more generalizable rewards by using a divide-and-conquer approach, focusing on learning features separately from learning how to combine them. We introduced feature traces as a novel type of human input that allows for intuitive teaching of non-linear features from high-dimensional state spaces. We then presented two instantiations of our FERL algorithm: one that enables expanding the robot's feature set in online reward learning situations, and one that lets the user sequentially teach every feature and then combine them into a reward. In extensive experiments with a real robot arm and a user study in simulation, we showed that online FERL outperforms deep reward learning from demonstrations (ME-IRL) in data-efficiency and generalization. Offline FERL similarly beats ME-IRL when the features used are of high enough quality, but the results are less conclusive when using very noisy features.

Implications for Online Reward Learning. Because they have to perform updates in real time from very little input, online reward learning methods represent the reward as a linear function of a set of hand-engineered features. Exhaustively choosing such a set a priori puts too much burden on system designers, and an incomplete set of features can lead to learning the wrong reward. Prior work enabled robots to at least detect that its feature space is insufficient to explain the human's input [43], but then the robot's only option was to either not update the reward or completely stop task execution. Our online FERL approach provides an alternative that allows people to teach features when the robot detects it is missing something, and then update the reward using the new feature set. Although we presented experiments where the robot learns rewards from corrections, our framework can conceivably be adapted to any online reward learning method, provided there is a way to detect the feature set is insufficient. Recent work on confidence estimation from human demonstrations [38] and teleoperation [313] offers encouraging pathways to adapting FERL to other online human-robot collaborative settings.

Implications for Learning Complex Rewards from Demonstrations. Reward learning from raw state space with expressive function approximators is considered difficult because there exists a large set of functions $r_\theta(s)$ that could explain the human input. For example, in the case of learning from demonstrations, many functions $r_\theta(s)$ induce policies that match the demonstrations' state expectation. The higher dimensional the state s, the more human input is needed to disambiguate between those functions sufficiently to find a reward r_θ that accurately captures human preferences. Without that, the learned reward is unlikely to generalize to states not seen during training and might simply replicate the demonstrations' state expectations. We presented evidence that offline FERL may provide an alternative to better disambiguate the reward and improve generalization.

The reason our divide-and-conquer approach can help relative to relying on demonstrations for everything is that demonstrations aggregate a lot of information. First, by learning features, we can isolate learning what matters from learning how to trade off what matters into a single value (the features vs. their combination) – in contrast, demonstrations have to teach the robot about both at once. Second, feature traces give information about states that are not on optimal trajectories, be it states with high feature values that are undesirable, or states with low feature values where other, more important features have high values. Third, feature traces are also structured by the monotonicity assumption: they tell us relative feature values of the states along a trace, whereas demonstrations only tell us about the aggregate reward across a trajectory. Thus, by focusing on learning features first before combining them into a reward, the robot can incorporate all three benefits and ultimately improve reward learning from demonstrations.

Limitations and Future Work. Our work is merely a step towards understanding how explicitly focusing on learning features can impact reward learning generalization and sample complexity. While FERL enables robots to learn features and induce structure in reward learning, there are several limitations that may affect its usability.

Our user study provides evidence that non-expert users can, in fact, use FERL to teach good features. However, due to the current pandemic, we conducted the study in a simulated environment instead of in person with the real robot, and most of our users had technical background. It is unclear how people without technical background would perform, and especially how kinesthetically providing feature traces (instead of clicking and dragging in a simulator) would affect their perception of the protocol's usability. Further, we only tested whether users could teach features we tell them about, so we still need to test whether users can teach features they implicitly know about (as would happen when intervening to correct the robot or designing a reward from scratch).

Even if people know the feature they want to teach, it might be so abstract (e.g. comfort) that they would not know how to teach it. Moreover, with the current feature learning protocol, they might find it cumbersome to teach discontinuous features like constraints. We could ease the human supervision burden by developing an active learning approach where the robot autonomously picks starting states most likely to result in informative feature traces. For instance, the robot could fit an ensemble of functions from traces online, and query for new traces from states where the ensemble disagrees [238]. But for such complex features, it may be more effective to investigate combining feature traces with other types of structured human input.

The quality of the learned rewards depends directly on the quality of the learned features. When the human provides feature traces that lead to good features, many of our experiments demonstrate that they induce structure in the reward learning procedure that helps generalization and sample complexity. However, if the robot learns features that are too noisy or simply incorrect, that (wrong) structure may impair performance. We saw an example of this when we tried to utilize the user study features for reward learning. In online FERL where a single feature was missing, the structure captured by the (noisy) non-expert features was still helpful in learning a better reward than the baseline. However, when trying to combine multiple noisy features in offline FERL, reward learning did not see a benefit. Future work must investigate ways in which the robot can determine whether to accept or reject the newly learned feature. One idea is to use our current framework's confidence estimation capability in Sec. 3.4.2.2 to determine whether the learned feature set explains the human's reward input. Another idea is to visualize either the feature function or examples of behaviors induced by it, and let the person decide whether the learned feature is acceptable.

While we show that FERL works reliably in 27D, more work is necessary to extend it to higher dimensional state spaces, like images. In our discussion in Sec. 3.9.2.3, we show how spurious correlations in large input spaces may affect the quality of the learned features in low data regimes. To counteract that, we could ask the person for more data, but after a certain point this becomes too burdensome on the user. Alternatively, approaches that encode these spaces to lower dimensional representations or techniques from causal learning, such as Invariant Risk Minimization [19], could help tackle these challenges.

3.9 Additional Details and Comparisons

3.9.1 Method Details

3.9.1.1 Incorporating Relative Values in Training. Concretely, given start state s_0, a relative value v_0 acts as a modifier for what $\phi_\psi(s_0)$ should be relative to ϕ_ψ's minimum value. If we consider the maximum feature value to be ϕ_ψ^{max} and the minimum one ϕ_ψ^{min}, we can define the feature range $\phi_\psi^{range} = \phi_\psi^{max} - \phi_\psi^{min}$. Then, v_0 shifts the desired feature value $\phi_\psi(s_0)$ in proportion to this range. When comparing $\phi_\psi(s_0)$ to the maximum value ϕ_ψ^{max}, their difference should be $\phi_\psi^{max} - \phi_\psi(s_0) = (1 - v_0) * \phi_\psi^{range}$. For example, if $v_0 = 0.3$, meaning the trace starts somewhere with a feature value 30% higher than the minimum, their difference is 70% of the feature range. If v_0 is the default 1, their difference becomes 0, meaning $\phi_\psi(s_0)$ is the maximum.

Similarly, a relative value v_n would also shift the feature value of an end state s_n in proportion to ϕ_ψ^{range}. This time, when comparing $\phi_\psi(s_n)$ to the minimum value ϕ_ψ^{min}, their difference will be $\phi_\psi(s_n) - \phi_\psi^{min} = v_n * \phi_\psi^{range}$. For example, if $v_n = 0.3$, meaning the trace ends somewhere with a feature value 30% higher than the minimum, their difference is 30% of the feature range. If v_n is the default 0, their difference becomes 0, meaning $\phi_\psi(s_n)$ is the minimum.

To incorporate the relative values v_0 and v_n into the training procedure, we have to use them to modify the feature values that the predictor in (3.3) is applied to. Given start states s_0 and s_0', instead of comparing $\phi_\psi(s_0)$ to $\phi_\psi(s_0')$ directly, we compare the altered feature values $\phi_\psi(s_0)' = \phi_\psi(s_0) + (1 - v_0) * \phi_\psi^{range}$ and $\phi_\psi(s_0')' = \phi_\psi(s_0') + (1 - v_0) * \phi_\psi^{range}$. As such, the training loss uses $P(\phi_\psi(s_0)' > \phi_\psi(s_0')')$ as a predictor. Similarly, given end states s_n and s_n', instead of comparing $\phi_\psi(s_n)$ to $\phi_\psi(s_n')$ directly, we compare the altered

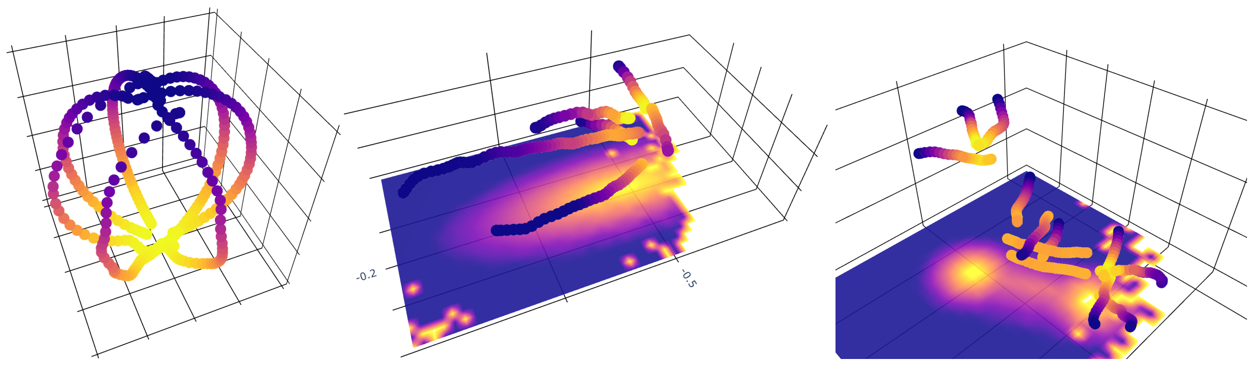


Figure 3.16: (Left) Feature traces for *coffee*. We show the xyz values of the x-axis base vector of the EE orientation. The traces start with the EE pointing downwards and move it upwards. (Middle) Feature traces for *proxemics* with the human at $xy = [-0.2, -0.5]$, with GT feature values projected on the xy-plane. Some traces are longer than others, to signal that the human dislikes the EE being in front of them more than to the sides. (Right) Feature traces for *between objects*, with GT feature values projected on the xy-plane. Notice a mix of traces teaching about the two objects and about the space between them.

feature values $\phi_\psi(s_n)' = \phi_\psi(s_n) - v_n * \phi_\psi^{range}$ and $\phi_\psi(s_n')' = \phi_\psi(s_n') - v_n * \phi_\psi^{range}$. As such, the training loss uses $P(\phi_\psi(s_n)' > \phi_\psi(s_n')')$ as a predictor.

3.9.2 Experimental Details

3.9.2.1 Protocols for Feature Trace Collection. In this section, we present our protocol for collecting feature traces for the six features discussed in Sec. 3.5.1. As we will see, the traces collected from the human only noisily satisfy the assumptions in Sec. 3.3.2. Nevertheless, as we showed in Sec. 3.5.1, FERL is able to learn high quality feature functions.

For *table*, the person teaches that being close to the table, anywhere on the xy plane, is desirable, whereas being far away in height is undesirable. As such, in Fig. 3.2 on the left traces traverse the space from up at a height, until reaching the table. A few different starting configurations are helpful, not necessarily to cover the whole state space, but rather to have signal in the data: having the same trace 10 times would not be different from having it once.

For *laptop*, as described in the text and shown in Fig. 3.2 on the right, the person starts in the middle of the laptop, and moves away a distance equal to the bump radius desired. Having traces from a few different directions and heights helps learn a more distinct feature. For *test laptop location*, the laptop's location at test time is not seen during training. Thus, the training traces should happen with various laptop positions, also starting in the middle and moving away as much distance as desired.

When teaching the robot to keep the cup upright (*coffee*), the person starts their traces by placing the robot in a position where the cup is upside-down, then moving the arm or rotating the EE such that it points upright. Doing this for a few different start configurations

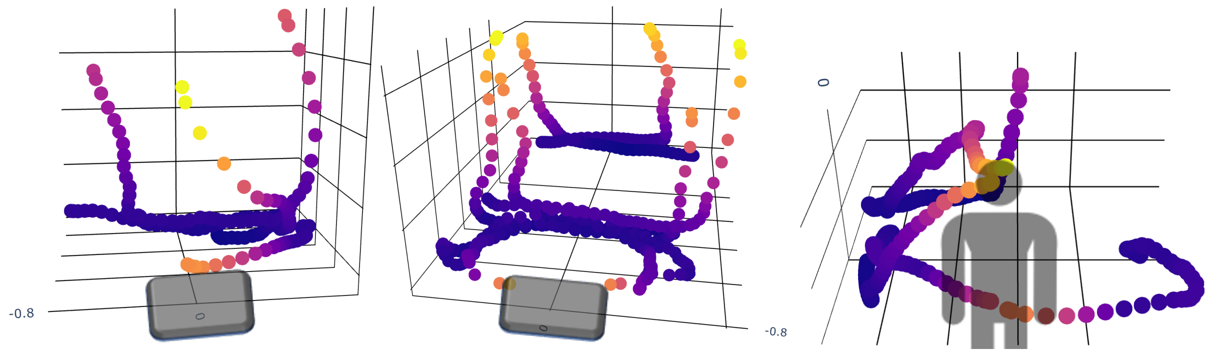


Figure 3.17: A few representative demonstrations collected for *Laptop Missing* (left), *Table Missing* (middle), and *Proxemics Missing* (right). The colors signify the true reward values in each task, where yellow is low and blue is high.

helps. Fig. 3.16 (left) shows example traces colored with the true feature values.

When learning *proxemics*, the goal is to keep the EE away from the human, more so when moving in front of their face, and less so when moving on their side. As such, when teaching this feature, the person places the robot right in front of the human, then moves it away until hitting the contour of some desired imaginary ellipsis: moving further in front of the human, and not as far to the sides, in a few directions. Fig. 3.16 (middle) shows example traces colored with the GT feature values.

Lastly, for *between objects* there are a few types of traces, all shown in Fig. 3.16 (right). First, to teach a high feature value on top of the objects, some traces need to start on top of them and move away radially. Next, the person has a few options: 1) record a few traces spanning the line between the objects, at different heights, and labeling the start and the end the same; 2) starting anywhere on the imaginary line between the objects and moving perpendicularly away the desired distance, and labeling the start; 3) starting on top of one of the objects, moving towards the other then turning away in the direction orthogonal to the line between the objects.

3.9.2.2 Protocols for Demonstration Collection. In an effort to make the ME-IRL comparison fair, we paid careful attention to collecting informative demonstrations for both reward learning settings in Sec. 3.6 and Sec. 3.7.

In the online setting, for each unknown feature, we recorded a mix of 20 demonstrations about the unknown feature only (with a focus on learning about it), the known feature only (to learn a reward weight on it), and both of them (to learn a reward weight combination on them). We chose diverse start and goal configurations to trace the demonstrations.

For *Laptop Missing*, we had a mix of demonstrations that start close to the table and focus on going around the laptop, ones that are far away enough from the laptop such that only staying close to the table matters, and ones where both features are considered. Fig. 3.17 (left) shows examples of such demonstrations: the two in the back start far away enough from the laptop but at a high height, and the two in the front start above the laptop at different heights.

For *Table Missing*, we collected a similar set of trajectories, although we had more demonstrations attempting to stay close to the table when the laptop was already far away. Fig. 3.17 (middle) shows a few examples: the two in the back start far away from the laptop and only focus on staying close to the table, a few more start at a high height but need to avoid the laptop to reach the goal, and another two start above the laptop and move away from it.

For *Proxemics Missing*, the most difficult one, some demonstrations had to avoid the person slightly to their side, while others needed to avoid the person more aggressively in the front. We also varied the height and start-goal locations, to ensure that we learned about each feature separately, as well as together. Fig. 3.17 (right) shows a few of the collected demonstrations.

In the offline setting, we took a similar approach to collecting demonstrations. For *One Feature*, we recorded 20 demonstrations starting far from the table and moving close to it, making sure to vary the start and end configurations. For Two Features, we collected 40 demonstrations (double the amount for two features) with a similar protocol to the *Laptop Missing* and *Table Missing* tasks in the online setting. Lastly, for the *Three Features* task we obtained 60 demonstrations, focusing on each feature separately, every pair of two, and the full combination of three features.

3.9.2.3 Raw State Space Dimensionality. Throughout our experiments, we chose a 36D input space made out of 27 Euclidean coordinates (xyz positions of all robot joints and environment objects) and 9 entries in the EE's rotation matrix. We now explain how we chose this raw state space, how spurious correlations across different dimensions can reduce feature learning quality, and how this adverse effect can be alleviated.

First, note that the robot's 7 joint angles and the xyz positions of the objects are the most succinct representation of the state, because the positions and rotation matrices of the joints can be determined from the angles via forward kinematics. With enough data, the neural network should be able to implicitly learn forward kinematics and the feature function on top of it. However, we found that applying forward kinematics a-priori and giving the network access to the xyz positions and rotation matrices for each joint improve both data efficiency and feature quality significantly. In its most comprehensive setting, thus, the raw state space can be 97D (7 angles, 21 xyz coordinates of the joints, 6 xyz coordinates of the objects, and 63 entries in rotation matrices of all joints).

Unfortunately, getting neural networks to generalize on such high dimensional input spaces, especially with the little data that we have access to, is very difficult. Due to the redundancy of the information in the 97D state space, the feature network ϕ_ψ frequently picks up on spurious correlations in the input space, which decreases the generalization performance of the learned feature. In principle, this issue could be resolved with more diverse and numerous data. Since we want feature learning to be as effortless as possible

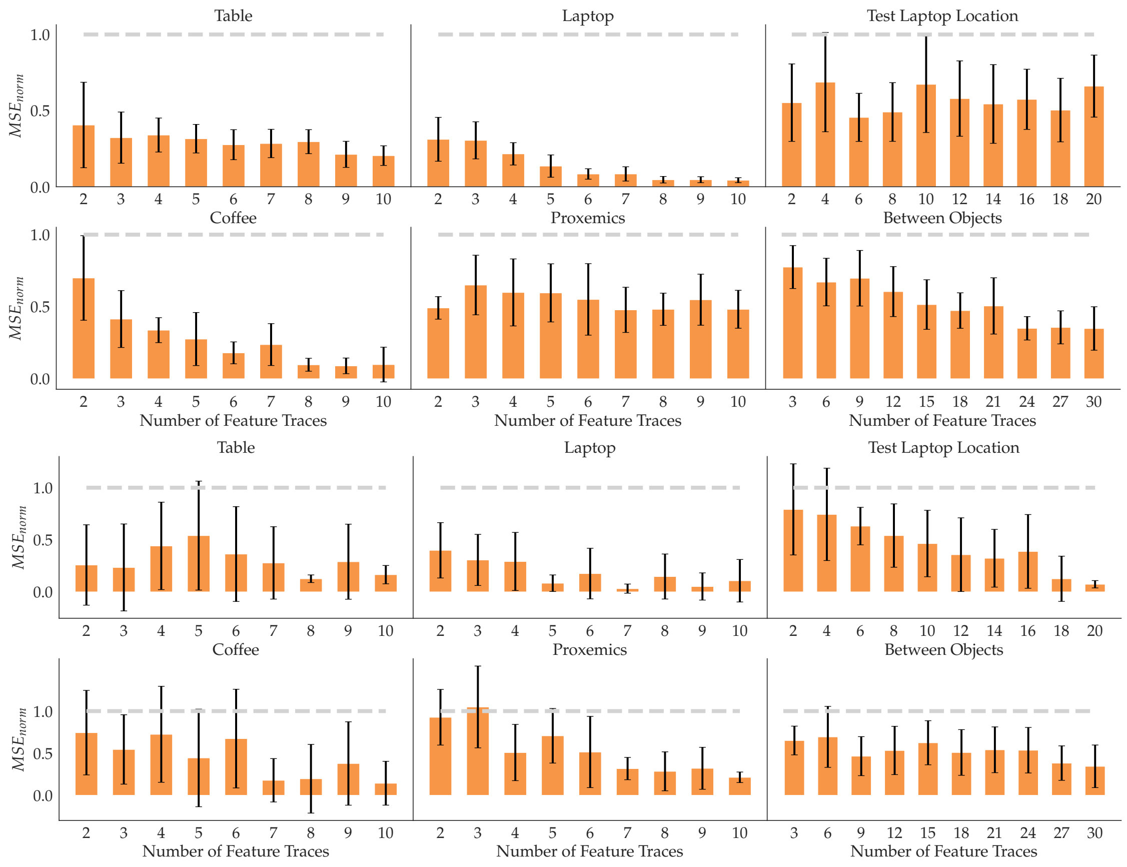


Figure 3.18: Quantitative feature learning results for 36D without (above) and with (below) the subspace selection heuristic. For each feature, we show the MSE_{norm} mean and standard error across 10 random seeds with an increasing number of traces (orange) compared to random performance (gray).

for the human, we instead opted for the reduced 36D state space, focusing directly on the xyz positions and the EE orientation.

Now, as noted in Sec. 3.5.1, the spurious correlations in the 36D space still made it difficult to train on both the position and orientation subspaces. To better separate redundant information, we devised a heuristic to automatically select the appropriate subspace for a feature. For each subspace, the algorithm first trains a separate network for 10 epochs on half of the input traces and evaluates its generalization ability on the other half using the FERL loss. The subspace model with the lower loss (better generalization) is then used for ϕ_ψ and trained on all traces. We found this heuristic to work fairly well, selecting the right subspace on average in about 85% of experiments.

To test how well it works in feature learning, we replicated the experiment in Fig. 3.4 on the 36D state space, both with and without the subspace selection heuristic. A first obvious observation from this experiment is that performing feature learning on separate subspaces (Fig. 3.4) results in lower MSEs for all features and N number of traces than learning from all 36 raw states (Fig. 3.18). Without the heuristic (Fig. 3.18 above), we notice that, while spurious correlations in the raw state space are not problematic for some features (*table*, *coffee*, *laptop*, *between objects*), they can reduce the quality of the learned feature significantly for *proxemics* and *test laptop location*. Adding our imperfect heuristic (Fig. 3.18 below) solves this issue, but increases the variance on each error bar: while our heuristic can improve learning when it successfully chooses the correct raw state subspace, feature learning worsens when it chooses the wrong one.

In practice, when the subspace is not known, the robot could either use this heuristic or it could ask the human which subspace is relevant for teaching the desired feature. While this is a first step towards dealing with correlated input spaces, more work is needed to find more reliable solutions. A better alternative to our heuristic could be found in methods for causal learning, such as Invariant Risk Minimization [19]. We defer such explorations to future work.

3.9.2.4 User Study Instructions. The familiarization phase of the user study is crucial for making sure our participants are equipped to provide pedagogic feature traces. To properly train our participants, we provided them with an instruction video and a user manual prior to the study. The manual outlined the task they were going to be trained on (*human*), how features are visualized in the study, how to utilize the simulator interface, how to give feature traces in practice, and provided visual examples of traces that lead to high quality and low quality teaching. The user video essentially followed the outline of the manual, but we found that it provided a more practical illustration of the interface and the teaching procedure. If interested in the instruction video, see `https://youtu.be/y36hhb9DI24`.

During the study, the familiarization phase was 10 minutes long and it gave participants the opportunity to try out the instructions from the manual in practice. First, a window appears visualizing the feature on 10,000 states sampled in the reachable set of the robot. We explain verbally what the feature represents and how that definition ties into the 3D visualization. This step was crucial to ensure that all participants have a standardized understanding on the features they teach. After closing this window, the simulator interface opens up for training the feature. Because this was a familiarization phase, we guided the participants through the steps, answered questions about the simulator interface, and explained how to give diverse and pedagogic feature traces. Once the algorithm trained the *human* feature, the 3D visualization of the learned feature along the given traces appeared. We walked the participants through what went right and wrong in their teaching, and explained how they could have improved their traces. We offered them the opportunity to try again, but all users chose to begin the study. Once the second phase of the study began, we offered participants no feedback on their teaching.

3.9.3 Implementation Details

We report details of our training procedures and hyperparameters. We tried a few different settings, and we present the settings that worked best for each method. The code can be found at https://github.com/andreea7b/FERL.

3.9.3.1 Feature Learning Training Details. The feature function $\phi_\psi(s)$ is approximated by a 2 layer, 64 hidden units neural network. We used a leaky ReLu non-linearity for all but the output layer, for which we used the softplus non-linearity. We normalized the output of $\phi_\psi(s)$ every epoch by keeping track of the maximum and minimum output logit over the entire training data. Following the description in Sec. 3.3.2, the full dataset consists of $|\mathcal{T}| = \sum_{i=1}^{N} \binom{(n^i+1)}{2} + 2\binom{N}{2}$ tuples, where the first part is tuples encoding monotonicity and the second part is tuples encouraging indistinguishable feature values at the starts and ends of traces. Since $\sum_{i=1}^{N} \binom{(n^i+1)}{2} >> 2\binom{N}{2}$, in the dataset there are significantly fewer tuples of the latter than the former type. This imbalance can lead to the training converging to local optima where the start and end values of traces are significantly different across traces. We addressed this by using data augmentation on the equivalence tuples (adding each tuple 5 times) and weighing the loss L_{equiv} by a factor of 10, i.e. we picked $\lambda = 10$ in (3.6). We optimized our loss function using Adam for $K = 100$ epochs with a learning rate and weight decay of 0.001, and a batch-size of 32 over all tuples.

3.9.3.2 Online FERL Details. In the Online FERL implementation of Alg. 3, the robot uses TrajOpt [255] to plan a path from the start to the goal configuration using the initial parameters θ, then starts tracking it. When a person applies a correction, the robot records the instantaneous deviation at 100Hz frequency until the interaction concludes. Then, the robot uses the first of these deviations to estimate the confidence $\hat{\beta}$ in its ability to explain the push. If the robot needs to learn a new feature, i.e. $\hat{\beta} < \epsilon$, it pauses trajectory execution. We used $\epsilon = 0.1$ but we did not perform extensive parameter tuning.

After potentially learning the feature using Alg. 1, the robot uses its recorded sequence of instantaneous deviations to update θ and replan ξ. If the robot did learn a new feature, by now its configuration has changed as a result of collecting feature traces, so we place it at the last recorded configuration before feature learning happened, then resume new trajectory execution. More details on estimating $\hat{\beta}$, deforming the trajectory ξ by the correction, and parameters for updating θ can be found in Sec. 7.8.

3.9.3.3 Offline FERL Details. We optimize the loss in Alg. 2 with stochastic gradient descent using a learning rate of 1.0 and $K = 50$ iterations. At each iteration we have to generate a set of near optimal trajectories D^θ for the current reward. To do so, we take the start and goal pairs of the demonstrations and use TrajOpt [255] to generate an optimal trajectory for each start-goal pair, hence $|D^*| = |D^\theta|$. At every iteration, we estimate the gradient using the full batch of $|D^*|$ demonstration tuples.

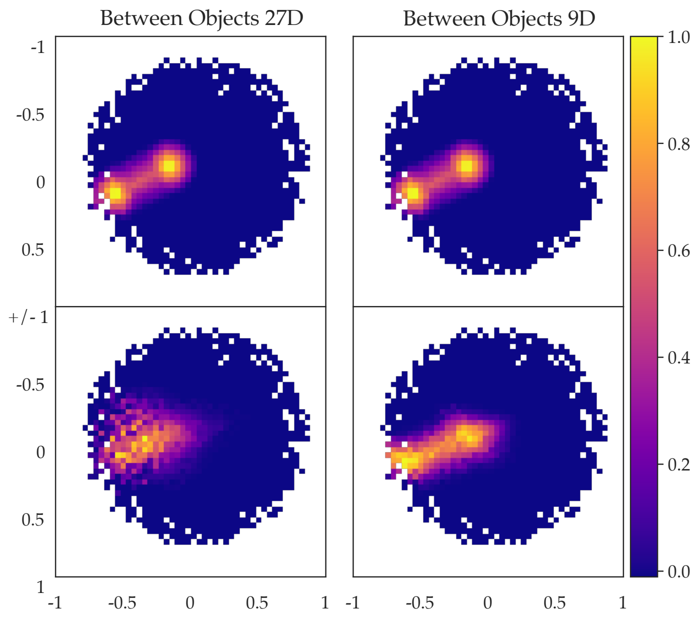


Figure 3.19: The *between objects* feature. (Left) Using a 27D highly correlated raw state space (xyz positions of all robot joints and objects), the learned feature (Down) does not capture the fine-grained structure of the ground truth (Up). (Right) When using only 9D (xyz positions of the EE and objects), the quality of the learned feature improves.

3.9.3.4 ME-IRL Training Details. We approximate the reward $r_\omega(s)$ by a 2 layer, 128 hidden units neural network, with ReLu non-linearities. In the online reward learning experiments in Sec. 3.6.1, we also add the known features to the output of this network before linearly mapping them to $r_\omega(s)$ with a softplus non-linearity. While D^* is given, at each iteration we have to generate a set of near optimal trajectories D^ω for the current reward $r_\omega(s)$. To do so, we take the start and goal pairs of the demonstrations and use TrajOpt [255] to generate an optimal trajectory for each start-goal pair, hence $|D^*| = |D^\omega|$. At each of the 50 iterations, we go through all start-goal pairs with one batch consisting of the D^* and D^ω trajectories of one randomly selected start-goal pair from which we estimate the gradient as detailed in Sec. 3.6.1. We optimize the loss with Adam using a learning rate and weight decay of 0.001.

3.9.4 Additional Results

3.9.4.1 *Between Objects* with 9D State Space. In Fig. 3.3 we saw that for *between features*, while FERL learned the approximate location of the objects to be avoided, it could not learn the more fine-grained structure of the ground truth feature. This could be an artefact of the spurious correlations in the high dimensional state space. To analyze this result, we trained a network with only the dimensions necessary for learning this feature: the xyz positions of the EE and of the two objects. The result in Fig. 3.19 illustrates that, in fact, our

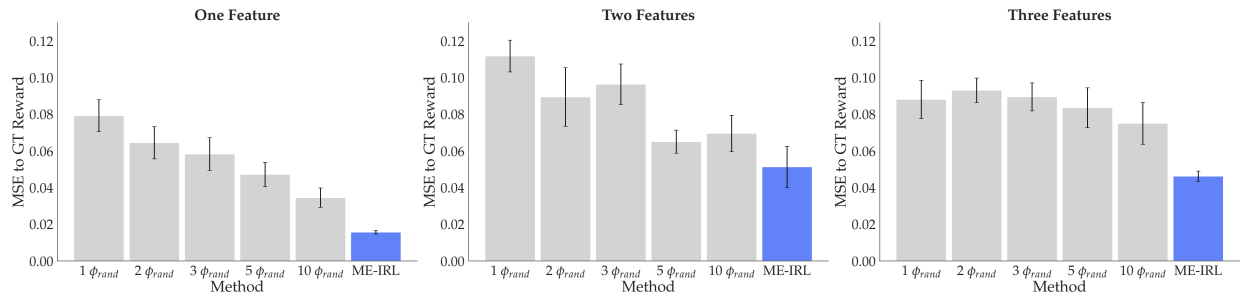


Figure 3.20: MSE of shallow ME-IRL with 1, 2, 3, 5, and 10 random features (gray) and deep ME-IRL (blue) to GT reward for *One Feature* (Left), *Two Features* (Middle), and *Three Features* (Right). The deep ME-IRL variant outperforms the shallow one.

method is capable of capturing the fine-grained structure of the ground truth; however, more dimensions in the state space induce more spurious correlations that decrease the quality of the features learned.

3.9.4.2 Baseline Comparison. Throughout our reward learning experiments in Secs. 3.6 and 3.7, we compare FERL to a deep implementation of ME-IRL. Here, we investigate using a shallow variant as a baseline instead, where the reward is modeled as a linear combination of random features ϕ_{rand}. We model each random feature as frozen randomly initialized neural networks with 2 layers and 256 units. This comparison should tell us whether using a deep architecture for reward learning provides an advantage when compared to learning rewards on top of random transformations of the input space.

For this experiment, we looked at three tasks where the GT reward is increasingly complex in the number of features: *One Feature*, *Two Features*, and *Three Features* from Sec. 3.7. We compare deep ME-IRL to a shallow implementation that has access to 1, 2, 3, 5, or 10 random features in the linear layer. We use as a metric the same *Reward Accuracy* metric as in Secs. 3.6 and 3.7, which computes the MSE between the normalized learned rewards and the GT.

Fig. 3.20 illustrates the differences between the 5 shallow variants (gray) and the deep ME-IRL (blue). For *One Feature*, the easiest case where the reward relies only on the *table* feature, we see that increasing the number of random features does improve the performance, but never beyond that of deep ME-IRL. The same trend disappears in the other, more complex, two cases. Overall, the deep variant consistently outperforms shallow ME-IRL with any of the tested number of random features, so we chose deep ME-IRL as the representative baseline in our main experiments.

Chapter 4

Learning Perceptual Features

This chapter is based on the paper "Learning Perceptual Concepts by Bootstrapping from Human Queries" [42], written in collaboration with Chris Paxton, Wei Yang, Balakumar Sundaralingam, Yu-Wei Chao, Maya Cakmak, and Dieter Fox.

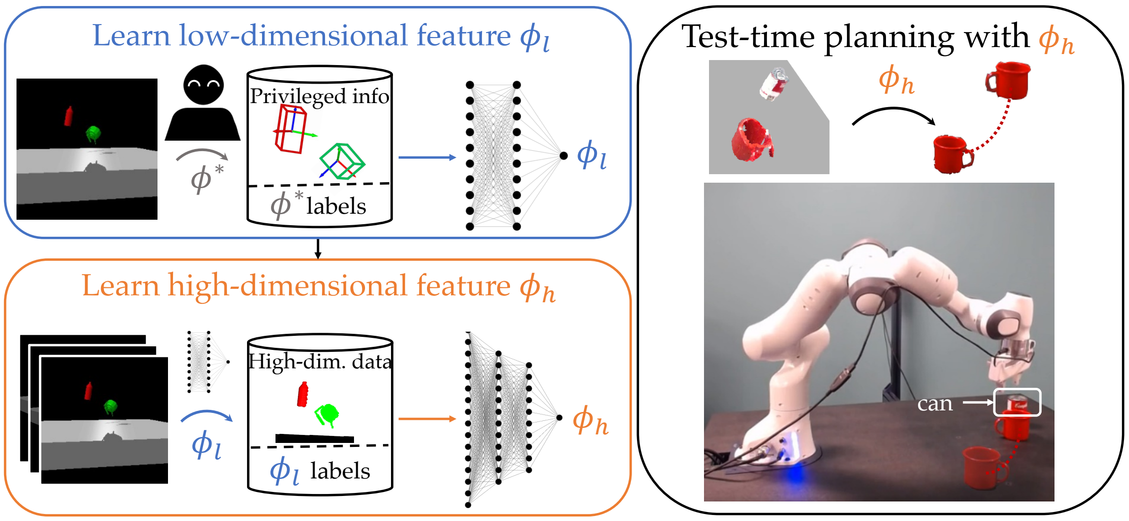

Learn low-dimensional feature ϕ_l
ϕ^*
Privileged info
ϕ^* labels
ϕ_l
Learn high-dimensional feature ϕ_h
ϕ_l
High-dim. data
ϕ_l labels
ϕ_h
Test-time planning with ϕ_h
ϕ_h
can

Figure 4.1: (Left) We propose a new approach for learning individual features whereby the robot collects labels from the human about the feature ϕ^* (e.g. *near*) and learns a low-dimensional feature ϕ_l on the *privileged information* space (e.g. poses and bounding boxes) (top), then uses ϕ_l to label data necessary for learning the high-dimensional feature ϕ_h (bottom). (Right) At test time, the robot can directly use the learned feature ϕ_h to produce a plan for moving the mug to be *near* the can. Additional qualitative results available at https://sites.google.com/nvidia.com/active-concept-learning.

In Chapter 3, we demonstrated how people can teach robots features of their representations by using a novel type of representation-specific input – feature traces – whose inherent structure can make training the feature function substantially more data efficient. In this chapter, we don't develop new types of feedback; instead, we turn our attention to the training process itself and focus on improving efficiency there.

The motivation in this section is similar to before: as robots are increasingly expected to perform tasks in human environments, from helping with household chores to cleaning up the office, we need to align their performance with the end user's unique needs. This means that the person should be able to teach their robot new *concepts* they care about – features expressing object-centric properties of the environment. Just like in previous chapters, a feature here maps the environment state to a value indicating how much the object-centric property is expressed, and the robot can optimize it to perform the person's desired task. For example, in Fig. 4.1 the user wants the robot to tidy the tabletop by moving the mug near the can. To accomplish this, the robot first learns the concept of what it means for objects to be *near* each other, and then moves the mug closer to the can.

A learned feature must operate on an input space that the robot understands. In Chapter 3, our experiments assumed that the environment state is fully observable and, thus, the input space directly consists of the state (robot poses, object poses, etc.). This assumption is convenient, but impractical: in the real world, robots rely on observations from sensors (e.g. images or point clouds). Since these observations are often high-dimensional, learning a perceptual feature that is robust across the input space is very challenging. Classical methods simplify this problem by relying on a pre-processing step, extracting geometric information like object poses and bounding boxes from the high-dimensional sensor data [204, 275, 208]. The human can then teach a feature mapping from this lower dimensional space, for instance by labeling when objects are near or far for the *near* example in Fig. 4.1. By transforming the sensor input into a lower dimensional space, the robot can learn the feature quickly even from limited human input.

Unfortunately, recovering accurate geometries from real-world sensor data is challenging: even modern pose estimators [295, 271, 83, 284] struggle when confronted with partial occlusions or novel objects [154]. Instead, recent deep learning alternatives learn perceptual features directly from the sensor data, without any pre-processing [224, 303]. These methods are usually trained in simulation, where a variety of objects can be manipulated in diverse configurations, resulting in better generalization than classical approaches [154, 211, 303]. However, because of the high dimensionality of the input space, the robot needs unreasonably many human-labeled examples, making a new feature impractical and cumbersome for a human to teach.

We propose getting the best of both worlds: learn features from high-dimensional sensor data with limited human labeling effort. We observe that, while the robot only has access to the high-dimensional sensor inputs during task execution, at training time it has a simulator containing *privileged information* akin to the geometries that classical approaches tried to compute. In Fig. 4.1, this privileged space is the object poses and bounding boxes – a much simpler representation than their high-dimensional point cloud equivalent. Our idea is to learn a low-dimensional, geometric variant of the feature on the privileged space, then treat it as a labeler and use it to automatically label high-dimensional data in the simulator (Fig. 4.1). This lets us generate a large, diverse, and *automatically labeled* dataset for training a high-dimensional, perceptual feature – or concept – which can be directly applied to real-world settings without additional human input.

Since these low-dimensional spaces result in faster training and are often semantically meaningful, they allow for richer human interaction. We thus investigate three types of human input for feature learning (demonstration, label, and feature queries), and propose an active learning strategy for informative queries. We then showcase our framework called Perceptual Concept Bootstrapping (PCB) in experiments both in simulation and on a real Franka Panda robot.

4.1 Prior Work

Feature Learning from Low-dimensional Geometries. Traditionally, features are hand-engineered by the system designer prior to robot deployment [311, 2, 123]. Unfortunately, by relying entirely on prior specification, the robot cannot adapt its task execution to an end user's needs. Recent works address this problem by allowing the robot to either infer features from task demonstrations [68, 177] or learn them directly from the human [40, 41] (as we did in Chapter 3). While these methods enable the robot to learn after deployment, they have been primarily demonstrated on low-dimensional spaces.

We can bypass the high-dimensional learning problem by extracting low-dimensional geometric information and learning a feature function on top of it [204, 275, 208]. However, recovering accurate geometries from real-world sensor data is challenging: even modern pose estimators [295, 271, 83, 284] struggle with the partial occlusions or novel objects that appear in open-world environments [154]. As such, we seek to learn features that operate directly on high-dimensional input, without any intermediary pre-processing.

Learning from High-dimensional Sensor Data. Deep learning handles high-dimensional data by using a function approximator to learn low-dimensional embeddings, hoping to capture salient aspects of the environment. Deep inverse reinforcement learning (IRL) and imitation learning approaches, in particular, use demonstrations to automatically extract behavior-relevant representations [96, 291, 259]. Unfortunately, to work reliably on high-dimensional inputs and generalize outside of the training distribution, these methods require a large amount of data from the human [102, 241].

Recent work in the auto-encoder community suggests that we can improve data efficiency by learning a disentangled latent space from weakly labeled examples of many features [140]. However, this approach still requires the user to label tens of thousands of examples for training. Moreover, while these methods are aligned with our goals of capturing important aspects of robotic tasks, they are complementary in that they focus on learning latent embeddings of the high-dimensional data, not representations made out of individual, meaningful perceptual features.

Feature Learning from High-dimensional Sensor Data. Instead of learning a universal representation, other work learns specific features directly from high-dimensional data [224, 303]. In particular, these methods learn from segmented object point clouds, which are easy to obtain and have successfully been used in other perception pipelines [112]. The disadvantage of this approach is that it still requires large amounts of data (thousands of labeled examples), making it unsuitable for learning the feature from a human. We look at how we can teach similar perceptual features from high-dimensional point cloud data, but do so quickly and efficiently with the help of the privileged space.

4.2 Approach: Perceptual Feature Bootstrapping

Our goal is to enable humans to teach robots perceptual features operating in high-dimensional input spaces, like segmented object point clouds. We assume that the robot may query the human for labeled examples of the desired feature, but we wish to learn the feature with as few human labels as possible. As training high-dimensional features is data intensive [224, 303], we propose to learn the feature first in a simpler, lower dimensional input space, then use it to label as much high-dimensional data as needed to train the feature in the target high-dimensional sensor space.

4.2.1 Preliminaries

Formally, a feature is a function mapping from input state to a scalar, $\phi(s) : \mathbb{R}^d \rightarrow [0, 1]$, indicating how much feature ϕ is expressed at d-dimensional state $s \in \mathbb{R}^d$. In our setting, we assume the human teacher already knows the ground truth feature ϕ^* and can answer queries about it.

At training, the robot has a simulator that gives it access to the entire state s, but at test time it receives high-dimensional observations $o_h \in \mathbb{R}^h$ given by a transformation of the state $\mathcal{F}(s) : \mathbb{R}^d \rightarrow \mathbb{R}^h$. In the example in Fig. 4.1, s captures the objects' pose, mesh, color, etc, whereas o_h is only the segmented point cloud of the scene from a fixed camera view. The robot seeks to learn a high-dimensional feature mapping from these observations, $\phi_h(o_h) : \mathbb{R}^h \rightarrow [0, 1]$, so that it can use it in desired manipulation tasks later on.

To do so, we assume the robot can ask the person for state-label examples $(s, \phi^*(s))$, forming a data set $\{s, o_h, \phi^*(s)\} \in \mathcal{D}_\phi$. Since the high-dimensional observation o_h directly corresponds to state s, this data set has the crucial property that the same label $\phi^*(s)$ applies to both s and o_h:

$$\phi^*(s) = \phi_h(o_h), \forall s, o_h = \mathcal{F}(s) \ . \tag{4.1}$$

From here, one natural idea to learn ϕ_h is to treat it as a classification or regression problem and directly perform supervised learning on $(o_h, \phi^*(s))$ pairs. Unfortunately, to learn a meaningful decision boundary, this approach would require very large amounts of data from the person, making it impractical to have a user teach a new feature.

Instead, we assume the robot can use *privileged information* from the simulator as a low-dimensional observation $o_l \in \mathbb{R}^l$ given by a transformation $\mathcal{G}(s) : \mathbb{R}^d \rightarrow \mathbb{R}^l$. We think of this information as privileged because the robot has access to it during training but not at task execution time. In Fig. 4.1, o_l only needs the object poses and bounding boxes to determine whether the objects are near. The set of collected human examples then includes the low-dimensional observation: $\{s, o_l, o_h, \phi^*(s)\} \in \mathcal{D}_\phi$, which allows the robot to learn a low-dimensional variant of the feature, $\phi_l(o_l) : \mathbb{R}^l \to [0, 1]$, by extending the property in (4.1):

$$\phi^*(s) = \phi_h(o_h) = \phi_l(o_l), \forall s, o_h = \mathcal{F}(s), o_l = \mathcal{G}(s) \ . \tag{4.2}$$

We hypothesize that learning the low-dimensional feature ϕ_l from privileged information should require less human input than learning ϕ_h directly from high-dimensional data. Moreover, (4.2) allows the learned ϕ_l to act as a labeler, bypassing the need for additional human input. We, thus, break down the feature learning problem into two steps: leverage the human queries to learn a low-dimensional feature ϕ_l, then use it to ultimately learn the original high-dimensional ϕ_h.

4.2.2 Learning a Low-dimensional Feature

To learn ϕ_l the robot first needs to ask the human for $\mathcal{D}_\phi$. To ensure the robot can learn the feature with little data, we want a query collection strategy that balances being informative and not placing too much burden on the human. We consider two types of input that are easy to provide and commonly used in the human-robot interaction (HRI) literature [57]: *demonstration queries* and *label queries*. Since users may struggle to label continuous values, we simplify the labeling to consist of 0 (negative) or 1 (positive) for low and high feature values. Note that despite the labels being discrete, they can still be used to learn a model that predicts continuous values.

Demonstration queries, or *demo queries*, involve creating a new scenario and asking the human to choose states s that demonstrate the feature and label them according to ϕ^*. This method requires an interface that allows the person to directly manipulate the state of the environment and label it, like a simulator with keyboard or click control. For example, for the *near* feature in Fig. 4.1, the person could move the red object near the green one and label the state 1.0, symbolizing a high feature value. Here, the robot can only manipulate the constraints of the scenario (e.g. which objects are involved) and the human has complete control over the selection of the rest of the state (e.g. object poses).

If the human is pedagogic, demonstration queries provide the robot with an informative data set of examples that should allow it to learn the low-dimensional feature quickly. Unfortunately, this data collection method can be quite slow due to the fact that the person has to spend time both deciding on an informative state and manipulating the environment to reach it. This makes it challenging to use in data intensive regimes (like when training ϕ_h from the get-go) but ideal in the low-data ones we are interested in.

Label queries are a less time-consuming alternative where the robot synthesizes the full query state s, and the person simply labels it as 0 or 1. For instance, in Fig. 4.1 the robot picks both the objects and their poses. This type of query is much easier and faster for the person to answer, but places the burden of informative state generation entirely on the robot. Simply randomly sampling the state space might not be very informative for the desired feature. For example, for a feature like *above*, placing two objects at random locations will rarely result in examples where one object is above the other. As such, we need a way to select more useful queries.

We use active learning [238, 37], whereby the robot can proactively select query states that it deems more informative. Concretely, we interleave asking for a batch of query states with learning the feature ϕ_l^t from the t examples received so far. This way, the robot can use the partially-learned ϕ_l^t to inform the synthesis of a more useful next batch of queries. For every query, the robot chooses among three query synthesis strategies: 1. *random*: randomly generate a state $s \in \mathbb{R}^d$; 2. *confusion*: pick the state that maximizes confusion by being at $\phi_l^t(s)$'s decision boundary, i.e. $s = \arg\min_s(\|\phi_l^t(s) - 0.5\|)$; 3. *augment*: select a state that was previously labeled as a positive (or negative, whichever is rarer) and add noise to it. A *random* query serves as a proxy for exploring novel areas of the state space. In a simulator, this query can be generated by randomizing the parameters of the state (e.g. object meshes, poses, etc). The *confusion* query is a proxy for disambiguating areas of the state space where the current feature ϕ_l^t cannot determine whether the state has a positive or a negative feature value. The query state in this case is selected by optimizing the feature value to be 0.5 using the cross-entropy method [78, 159]. The *augment* query is useful for features where positives (or negatives) are rarer, like in the *above* example.

Active learning is possible when learning low-dimensional features because they have much shorter training cycles than their high-dimensional variants. Another advantage of low-dimensional spaces is that, while the transformation $\mathcal{F}$ cannot be modified because the robot is constrained to operate on o_h at test time, we have more flexibility over what $\mathcal{G}$ and o_l can be. We exploit this with a third type of human input called *feature queries* [57]. Feature queries typically involve asking the person whether an input space dimension is relevant for the target feature. However, this query is only useful if the dimension itself is meaningful to the human. We adapt feature queries and ask the person a few intuitive questions about the feature such that the answer informs the choice of the transformation $\mathcal{G}$. For example, a negative answer to the question "Does the size of the objects matter?" signals that o_l does not benefit from containing object bounding box information. These queries lead to an appropriate $\mathcal{G}$, which can further speed up the learning of ϕ_l.

Given a dataset of human labels $\mathcal{D}_\phi$, the robot can train a low-dimensional feature ϕ_l. We treat feature learning as a classification problem, approximate ϕ_l by a neural network, and train it on the $(o_l, \phi^*(s))$ examples in $\mathcal{D}_\phi$ using a binary cross-entropy loss.

4.2.3 Learning a High-dimensional Feature

Learning the perceptual feature requires a large amount of labeled high-dimensional data. Generating this data set is a two-step process: the robot needs to synthesize a large and diverse set of states s, which it then has to acquire labels for. However, as opposed to the low-dimensional case, this data set need not be directly labeled by the human: the learned low-dimensional feature itself can act as a labeler.

At training time the robot has a simulator, so we can randomly explore the state space for the data synthesis step. With the property in (4.2), we can use the low-dimensional feature ϕ_l to automatically label the states, generating the data set $\{s, o_l, o_h, \phi_l(o_l)\} \in \mathcal{D}_{\phi_l}$. To now learn the high-dimensional feature ϕ_h, we approximate it by a neural network and train it via classification on the $(o_h, \phi_l(o_l))$ examples in $\mathcal{D}_{\phi_l}$ using a cross-entropy loss.

4.2.4 Implementation Details

We used a multilayer perceptron (3 layers, 256 units) and a standard PointNet [289, 228] to represent the low- and high-dimensional features, respectively. Our features involved relationships between objects, so we represented the high-dimensional observation o_h with the relevant objects' segmented point clouds from the camera view, and the low-dimensional one o_l with object poses and bounding boxes.

For data generation, we modified the objects in the ShapeNet data set [61] to be aligned and scaled. We selected objects commonly found in tabletop manipulation tasks, like bowls, cereal boxes, cups, cans, mugs, bottles, cutlery, hammers, candles, teapots, fruit, etc. (Fig. 4.2). When synthesizing states $s \in \mathbb{R}^d$, we spawned pairs of two objects in the Isaac Gym simulator [200] and manipulated their poses, as well as the camera pose along the table plane. This process resulted in a variety of states with possibly occluded objects, from many camera views. Since our method allows us to generate as much simulated data as desired, our hope is to generalize to real-world conditions like other simulation-based methods do [224, 209, 294].

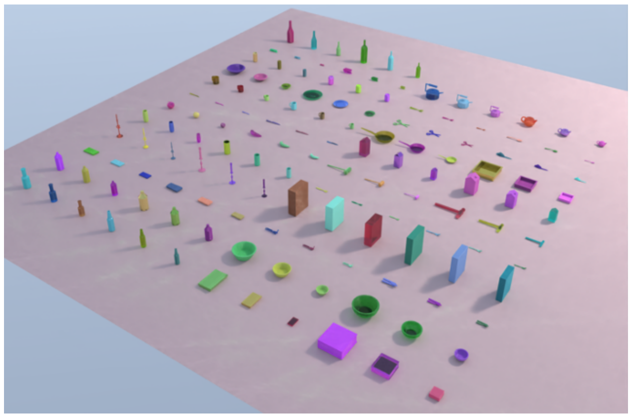


Figure 4.2: We show the aligned and scaled ShapeNet objects. We chose objects commonly found in manipulation tasks.

4.3 Experiments: Learning Perceptual Features

We now compare our label-efficient perceptual feature learning method PCB to a baseline that learns directly from high-dimensional input. PCB relies on a human-trained low-dimensional feature ϕ_l, for which we perform an extensive investigation in Sec. 4.4.

4.3.1 Experimental Design

Throughout our experiments, we synthesize queries by manipulating pairs of objects in the simulator: a stationary *anchor* and a *moving* object, which is related to the anchor by our feature. We investigate 9 spatial features:

1. *above*: angle between the objects' relative position and the world z-axis;
2. *above*$_{bb}$: intersection area of the two objects' bounding box projections on the world xy-plane;
3. *near*: inverse distance between the objects;
4. *upright*: angle between the moving object's z-axis and the world's;
5. *aligned*$_{horiz}$: angle between the objects' x-axes;
6. *aligned*$_{vert}$: angle between the objects' z-axes;
7. *forward*: angle between the anchor's x-axis and the objects' relative position;
8. *front*: angle between the anchor's x-axis and the objects' relative position;
9. *top*: angle between the anchor's z-axis and the objects' relative position.

For evaluation purposes, our ground truth feature implementations cut off the angles in *above*, *upright*, *aligned*$_{horiz}$, *aligned*$_{vert}$, *front*, and *top* after 45° and the distance in *near* after 0.3m, then normalize all feature values between 0 and 1. Fig. 4.3 showcases qualitative visualizations of the features.

Notably, some of the features involve object affordances (*upright*, *aligned*$_{horiz}$, *aligned*$_{vert}$, *forward*, *front*, *top*). For those, only a subset of the objects are applicable (e.g. a mug has a *front*, but a box doesn't), so we selected object subsets for each feature accordingly. The features *above*, *above*$_{bb}$, and *near* used all objects because they don't involve object affordances. For *upright* and *top*, we used objects with clear upright orientations: bottles, bowls, candles, mugs, cups, cans, milk cartons, pans, plates, and teapots. For *aligned*$_{horiz}$ we used objects that can be horizontally aligned: calculators, can openers, cutlery, hammers, pans, and scissors. For *aligned*$_{vert}$ we used objects that can be vertically aligned: bottles, boxes, candles, cups, milk cartons, and cans. For *forward* and *front* we used large objects with clear fronts: hammers, pans, and teapots. By default, the privileged space consists of the object poses, relative pose, positional difference, and bounding boxes.

4.3.2 PCB Results for Label Queries

We first show results with label queries. We compare PCB to a baseline that learns ϕ_h directly from the human queries, without an intermediate low-dimensional feature. For PCB, we chose to use the low-dimensional features ϕ_l trained using feature and label queries collected with the *confrand* and *augment* active learning strategies together. We show in Sec. 4.4 that this was the best performing ϕ_l with the overall cheapest type of human input. We use the features ϕ_l to label a large set of randomly generated 80,000

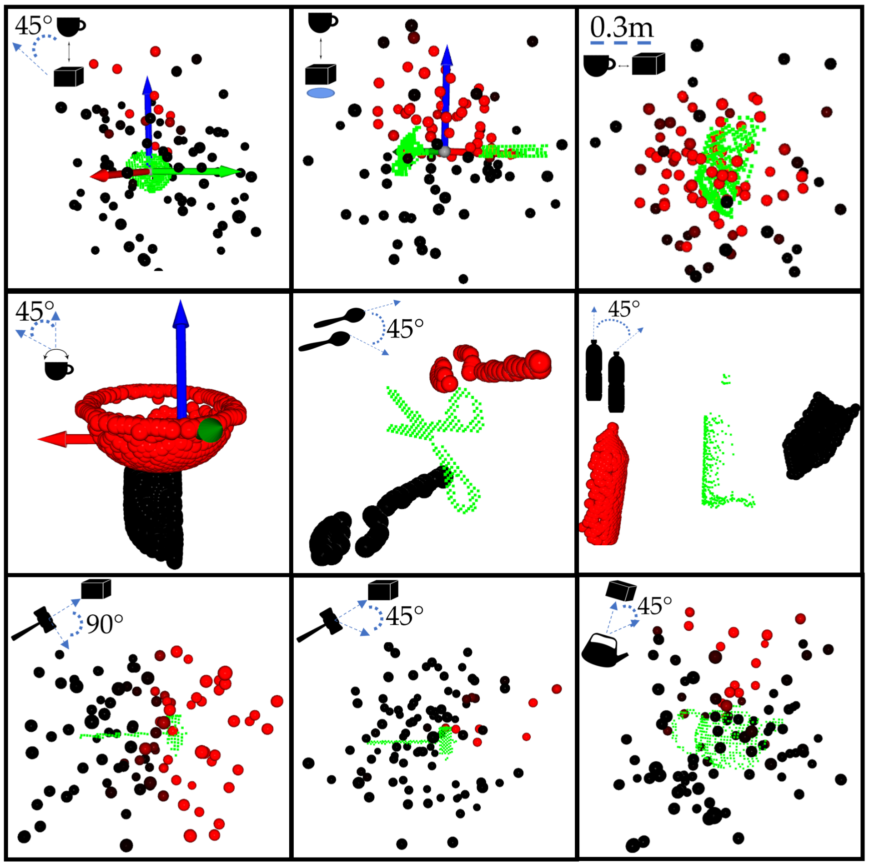


Figure 4.3: Visual representation of the 9 perceptual features learned with our method (icon in the top left of each box). The anchor (green) is joined by examples of the moving object represented as either partial object point clouds (middle: *upright*, *aligned*$_{horiz}$, *aligned*$_{vert}$) or object point cloud centers (top: *above*, *above*$_{bb}$, *near*; bottom: *forward*, *front*, *top*). We color predicted positive examples in red, and negative ones in black. For features defined with respect to the world coordinate frame, we additionally plot the frame.

training states, resulting in $\mathcal{D}_{\phi_l}$, then train ϕ_h using $\mathcal{D}_{\phi_l}$. Since the baseline is a PointNet that takes a long time to train, it is not suitable for active learning, so we train it with label queries generated with the *random* strategy. For additional comparison, we also train the baseline with the label queries collected by PCB with the *confrand* and *augment* strategies.

We train ϕ_h with each method and a varying number of queries, and report two metrics: 1) *Classification Accuracy*: how well the features can predict labels for a test set of states, and 2) *Optimization Accuracy*: how well the states induced by optimizing these features fare under the true ϕ^*. For *Classification Accuracy*, we use ϕ^* to generate a test set $\mathcal{D}_{test}$ of 20,000 state-label pairs such that they have an equal number of positives and negatives. This way, we probe whether the learned features perform well on both labels and don't bias to one. We measure ϕ_h's accuracy as the percentage of datapoints in $\mathcal{D}_{test}$ predicted correctly. For *Optimization Accuracy*, we sample 1,000 states $\mathcal{S}_{opt}$ with a feature

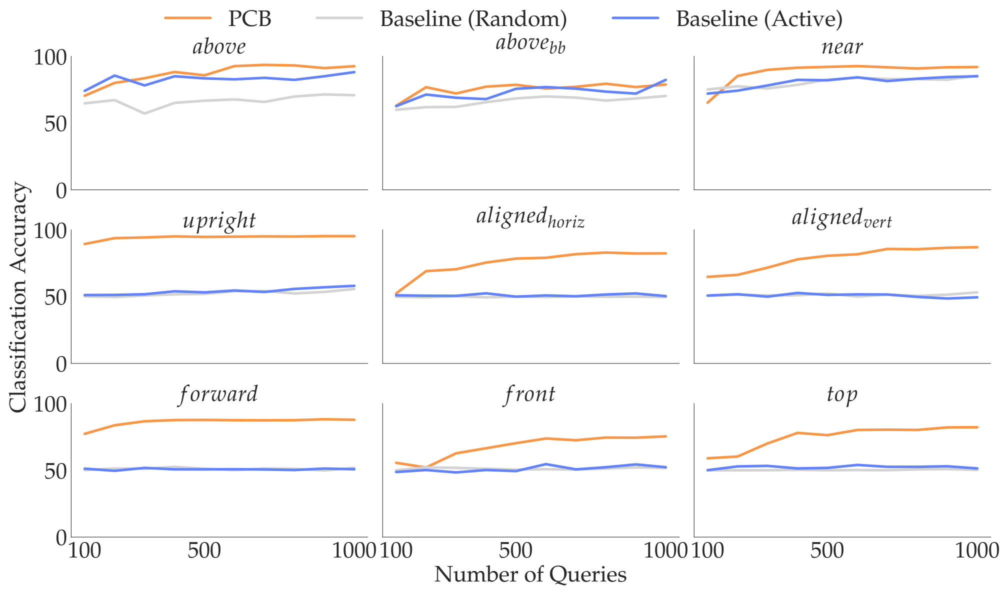


Figure 4.4: Classification accuracy on a held-out test data set, for models trained on a varying number of queries. PCB features (orange) correctly classify at least 80% of the data after the first 500 queries in most cases. The baseline with random queries (gray) struggles to perform better than random; when trained with PCB's actively collected data (blue), the baseline performs better for the simple features but fails on the last six features that involve affordances.

value of 0, and use the learned features to optimize them into a new set of states $\mathcal{S}'_{opt}$. We do so by finding a pose transform on the moving object that maximizes the feature value, and use the cross-entropy method [78]. Importantly, since this is happening at test time, we use the high-dimensional observation of the state o_h to perform the optimization. We evaluate $\mathcal{S}'_{opt}$ under ϕ^* and report the percentage of states that are labeled as 1. We present results for an arbitrarily chosen fixed seed.

4.3.2.1 Qualitative Results. Fig. 4.3 showcases the 9 features trained using PCB. For every feature, we show the anchor object in green (if applicable), together with positive and negative examples of the feature. For *above* and *above$_{bb}$* the positive examples above the anchor are sparse, which could make learning challenging from a data diversity perspective: if the robot doesn't query for enough positive examples, it won't be able to learn a meaningful decision boundary for these features. In contrast, *near* has a balanced mix of positives and negatives, making it a simpler feature to learn. The remaining six features all involve affordances which are dependent on the object shapes (e.g. a bowl is upright if its opening points upwards, an object is atop the kettle is it's placed above the lid, etc). Thus, to learn such features describing functionality across a plurality of objects and camera views, a method that learns directly from point clouds would need large amounts of data to accurately capture how the features relate to all objects' morphologies.

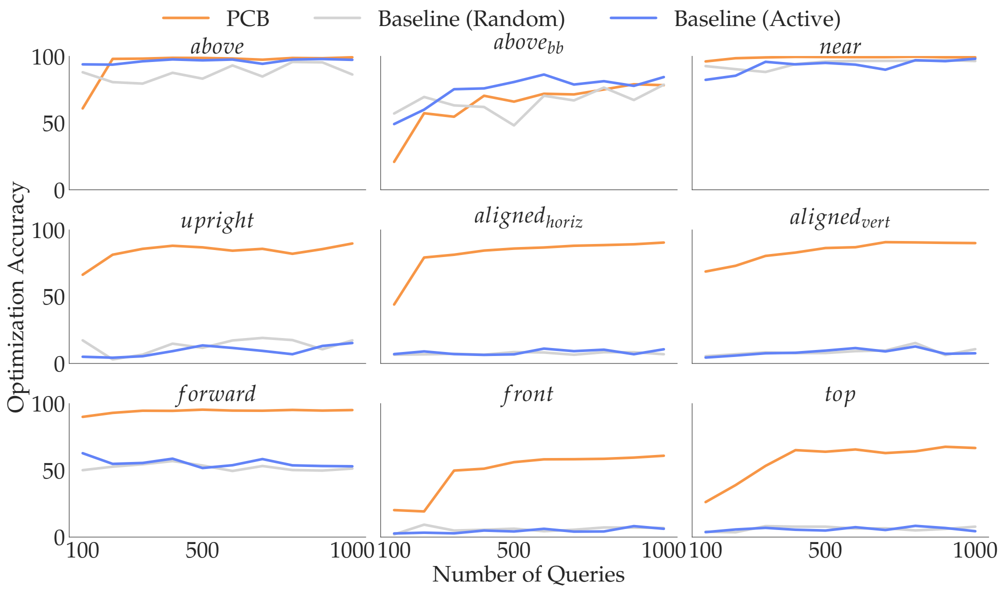


Figure 4.5: Accuracy when optimizing object poses based on the learned features (*Optimization Accuracy*). PCB (orange) produces satisfactory poses for most features, as opposed to the baseline (gray and blue) which sometimes cannot even surpass 10% performance.

The figure shows that PCB handles these challenges gracefully.

4.3.2.2 Quantitative Analysis. Fig. 4.4 shows *Classification Accuracy* results. Both variants of the baseline perform well for *above*, *above*$_{bb}$, and *near*, eventually reaching 70% performance. We think this happens because these features only require absolute position information, which is easy to infer from just the positions of the points clouds. The baseline trained on active data from PCB performs closer to PCB, suggesting that in some cases we may use the privileged information and low-dimensional features to guide the labeling process of high-dimensional data and be more sample efficient, without generating additional high-dimensional data. However, this does not always hold: the other six features involve affordances in addition to positions, which is much more challenging to capture with limited data. As a result, neither baseline version can achieve performance better than random. In contrast, PCB, which is able to generate thousands of high-dimensional training data points capturing different object point cloud morphologies, can successfully learn these kinds of features, correctly classifying at least 80% of the test data after the first 500 queries in most cases. In Fig. 4.5, *Optimization Accuracy* results tell a similar story. Our features can be optimized successfully with an accuracy of over 50%, meaning that we would be able to find positions for objects to satisfy these features [224]. Meanwhile, several baseline features have a success rate barely above 10%.

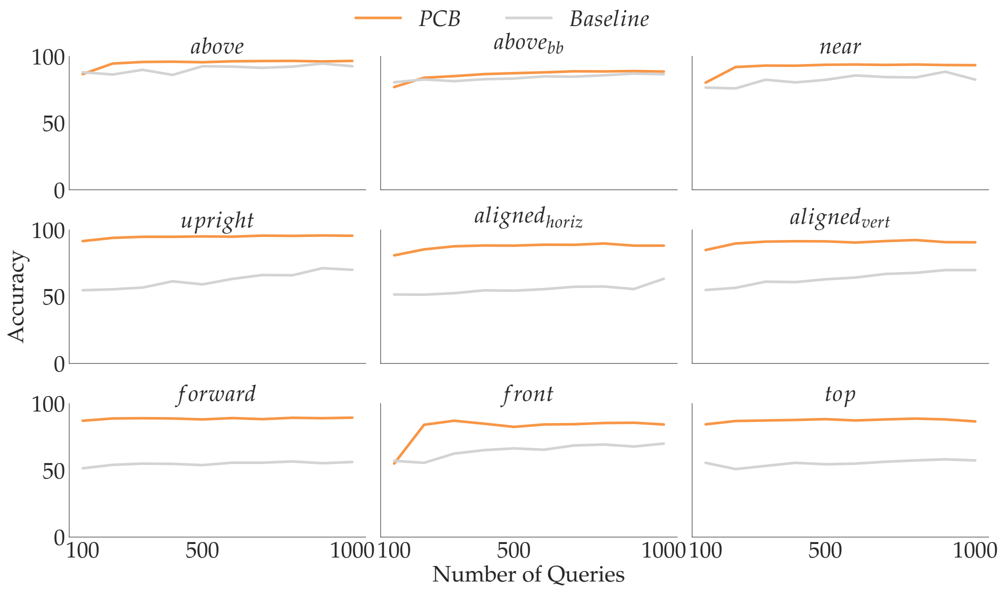


Figure 4.6: Classification accuracy on a held-out test data set (*Classification Accuracy*), for models trained on a varying number of queries. features trained using our PCB method (orange) correctly classify at least 80% of the test data after the first 200 demo queries. Meanwhile, the baseline (gray) struggles to perform better than random, especially on the last six features that involve affordances.

4.3.3 PCB Results for Demo Queries

We now expand on the results in Sec. 4.3.2 by showing the case where the human provides the robot with demonstration queries. We compare PCB to a baseline that learns ϕ_h directly from the queries. For PCB, we take the ϕ_l features we trained using both demonstration and feature queries in Sec. 4.4.1, and use them to label a large set of 80,000 training states, resulting in $\mathcal{D}_{\phi_l}$. Our method then trains ϕ_h using $\mathcal{D}_{\phi_l}$, while the baseline trains the same architecture using the original queries we used to learn ϕ_l. Importantly, both methods use well-balanced demonstration queries. We report results on the same two metrics from Sec. 4.3.2, *Classification Accuracy* and *Optimization Accuracy*.

Fig. 4.6 shows *Classification Accuracy* results. The baseline actually performs well for *above*, *above*$_{bb}$, and *near*, eventually reaching 80% performance. We think this happens because for these features it is easy to infer the necessary privileged information just from the positions of the point clouds. For example, for *near*, given the position of the two object point cloud centers, learning a relationship between their distance and the feature value should not require more than a few samples. The other features involve affordances in addition to position information, which is much more challenging to capture. As a result, the baseline can barely achieve performance better than random. In contrast, our method, which is able to generate thousands of high-dimensional training data points, can successfully learn these kinds of features, correctly classifying at least 80% of the test data

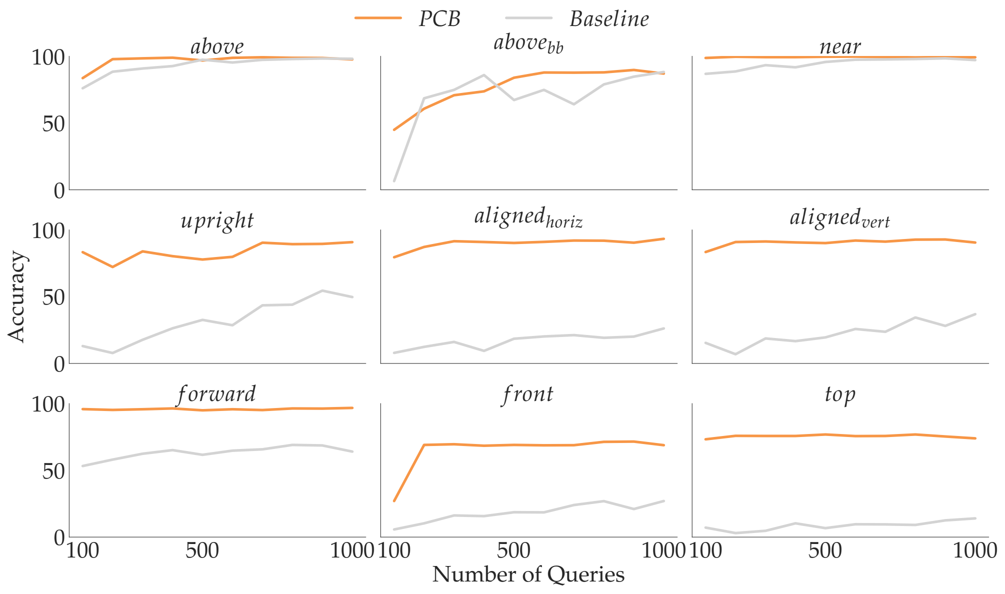


Figure 4.7: Accuracy when optimizing object poses based on the learned features (*Optimization Accuracy*). Our PCB method (orange) produces satisfactory poses for most features, as opposed to the baseline (gray) which sometimes cannot even surpass 25% performance.

after the first 200 queries. Note that PCB with demo queries reaches this accuracy faster than with the label queries from Sec. 4.3.2, but demo queries are more effortful to give than label queries. This shows the trade-off between human effort and informativeness we investigated in Sec. 4.4.

In Fig. 4.7, *Optimization Accuracy* results tell a similar story. Our features can be optimized successfully with an accuracy of over 50%, meaning that we would be able to find positions for objects to satisfy these features [224]. Meanwhile, several baseline features have a success rate barely above 25%.

4.4 Analysis: Learning Proxy Features for Bootstrapping

We saw how, given a low-dimensional feature learned from human input, our method can outperform the baseline learning directly from high-dimensional sensor data. We now analyze what are the best strategies for learning low-dimensional proxy features from human input. We seek to answer the following: **Q1:** Does querying via demonstration – the most informative type of query but also the most expensive – benefit learning when compared to random label queries? **Q2:** Does modifying the privileged information space via feature queries speed up learning? **Q3:** Can we choose label queries – the cheaper version of demo queries – that are more informative than random via active learning? **Q4:** How does labeling noise affect the quality of the learned features?

4.4.1 Benefits of Demonstration, Label, and Feature Queries

We first compare the three types of human queries across two dimensions: the query selection strategy and the privileged input space. For the former, while the robot could randomly synthesize states and ask the human to label them (i.e. random label queries), for some features such a strategy would rarely find states with positive feature values. In contrast, demonstration queries allow the human to select the states themselves, so they can balance the amount of positives and negatives the data set contains to be informative. As for the privileged input space, by default it contains many features that are correlated with one another or irrelevant to some features altogether. These redundant dimensions can make learning more difficult. Feature queries, with just a few simple and intuitive questions, can eliminate some dimensions of the input space that are unnecessary.

To answer **Q1** and **Q2**, we use a 2×2 factorial design. We manipulate the *query strategy* (*random label* and *demo*), and the *input space strategy* (*feature* and *no feature*). For both query strategies, we generate a dataset of labeled states as described in Sec. 4.2.4, and simulate human input by sampling examples randomly for *random label* or in a way that balances positives and negatives for *demo*. The practical difference is in the positives-to-negatives ratio: while for *random label* that may be low for certain features (other than *near* and *forward* the mean ratio is 0.08), for *demo* it is 1. For *feature*, we ask three intuitive questions: F1. Does the feature concern a single object? F2. Does the feature care about the objects' absolute poses or their relative one? F3. Do the object sizes matter? F1 discards dimensions from the redundant object (useful for features like *upright*). F2 gets rid of correlated features (absolute or relative pose). F3 drops bounding box information if the feature doesn't require it.

We compare the learned feature network ϕ_l to the ground truth ϕ^* with a metric similar to *Classification Accuracy* from Sec. 4.3: we measure ϕ_l's correct prediction rate for $\mathcal{D}_{test}$. In Fig. 4.8, we show results with varying amounts of queries from 100 to 1000. Comparing the solid lines, we immediately see that, for most features, demo queries perform much better than random label queries. The only features where this trend doesn't hold are *forward* and *near*, which are features where random sampling can already easily find many positives. This result stresses that having enough positives is crucial for learning good features. We can also compare the effect of feature queries: whether we use demo or label queries, feature queries considerably speed up learning, and this result holds across all 9 features. Another observation is that the combination of demo and feature queries plateaus in performance after about 200-300 queries, suggesting that, although each query requires more human time, the teaching process altogether might be shorter.

4.4.2 Active Query Labeling

We saw that demonstration queries substantially benefit feature learning when compared to random label queries. Unfortunately, demo queries are also very time-consuming

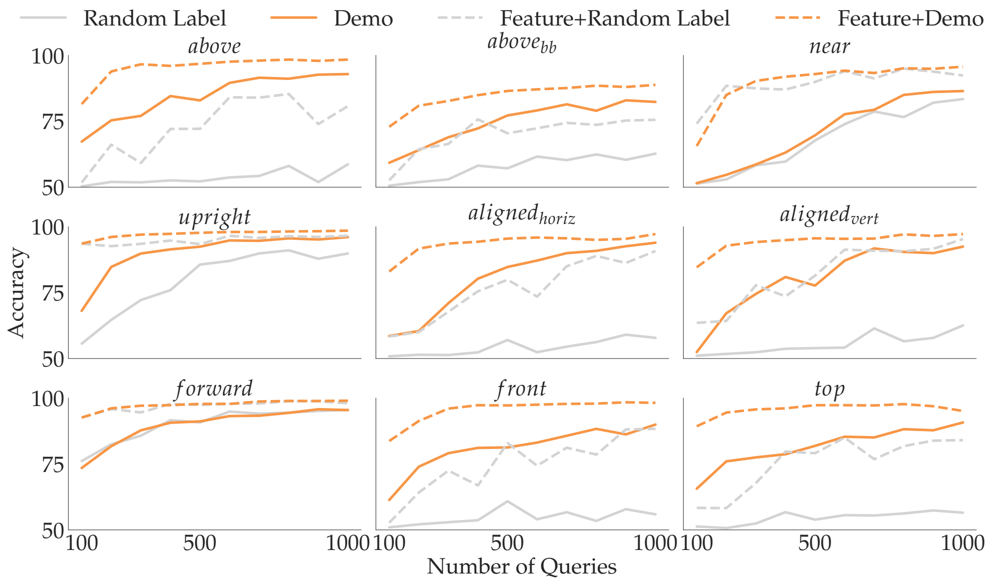


Figure 4.8: Comparing different query and input space strategies. *Demo* queries outperform *random label* for features with few positives, and *feature* queries improve learning speed.

to collect[1], which only makes them feasible in low-data regimes. In this section, we tackle **Q3** and explore whether we can make label queries more informative by employing active learning techniques, rather than simply randomly selecting them. This way, we can have the benefits of both informative query generation and easy label collection.

We use a 3×2 factorial design where we vary the *active strategy* (*random*, *confusion*, and *confrand*) and the *positives selection* (*augment* and *no augment*). As described in Sec. 4.2, *random* generates a query state randomly and *confusion* picks a state at the decision boundary of the currently learned feature. We also introduce *confrand*, which randomly selects between the two strategies, to balance exploration of new areas and disambiguation of the current feature. With an *augment* positives selection, for every query the method also randomly chooses whether to exploit the space of positives it has found so far or just go with the selected active strategy. We use a batch size of 100. We train ϕ_l with each strategy and varying number of queries, and report accuracy on $\mathcal{D}_{test}$.

In Fig. 4.9, we show results with increasing number of queries across the 6 total label query selection strategies. Right off the bat, we see that active learning helps more the harder it is to find positives. For features like *forward* or *near*, random label queries do well because the positive-to-negative ratio is already high. For all other features, however, active learning helps considerably, certain techniques more than others. A general trend is that using *augment* queries outperforms not using them, especially in lower data regimes, confirming our intuition that finding positives earlier on improves learning. Amongst

[1] An expert labels 100 queries in 2 minutes, but needs 10 minutes for the same amount of demo queries.

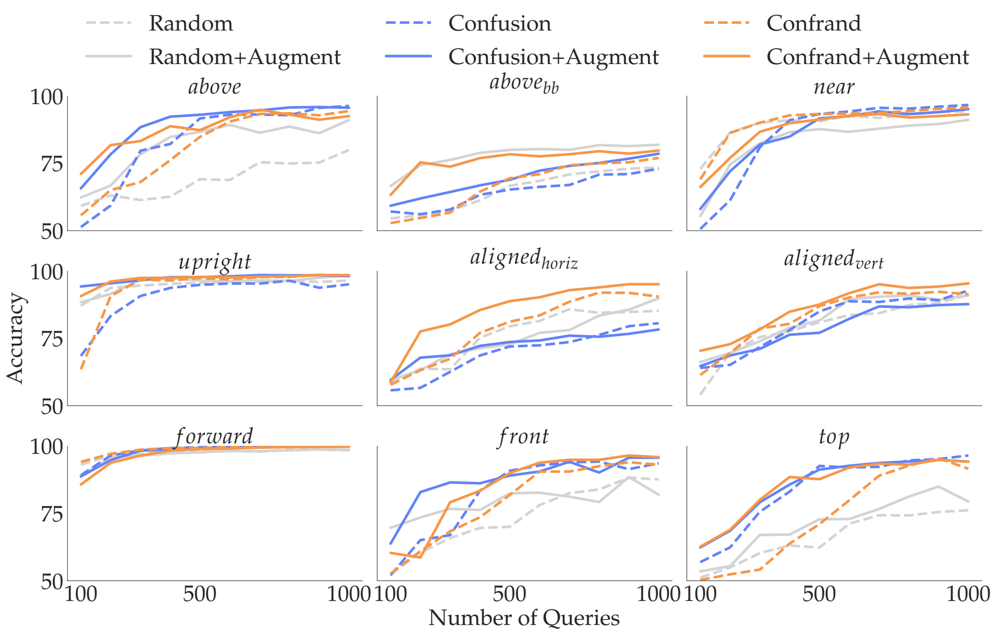


Figure 4.9: Comparison amongst active labeling and positives selection strategies. *Confrand* is the most consistently beneficial strategy, and *Augment* boosts performance, especially in low data regimes.

random, *confusion*, and *confrand*, we don't see a clear winner for all features, but *confrand*, the combination of novelty and uncertainty exploration, seems to perform the best across. It is encouraging to see that the performance can reach 80% accuracy after the first 500 queries, which would require a mere 10 minutes of human labeling time.

4.4.3 Noise Ablation in Human Query Labeling

So far we assumed the simulated human answered the queries perfectly. Since this won't necessarily be the case for novice users, we examine **Q4**, how labeling noise affects our feature learning results. We do this by varying the *noise level* by 0%, 1%, 5%, 10%, 25%, and 50%. A "noisy" query has its label flipped. 50% is equivalent to a random labeler. We train ϕ_l by adding varying noise levels to the queries and report accuracy on $\mathcal{D}_{test}$. Fig. 4.10 reveals that, unsurprisingly, the noisier the queries, the worse the learned feature performs. While unrealistic noise levels like 25% or 50% severely worsen the quality of the learned features, our method seems to be able to withstand lower noise levels.

4.5 Using Perceptual Features in Motion Planning Tasks

We test our learned PCB features ϕ_h on a Franka Panda robot with a RGB-D camera in motion planning tasks, as shown in Fig. 4.1. For each trial, a user selects the feature to test,

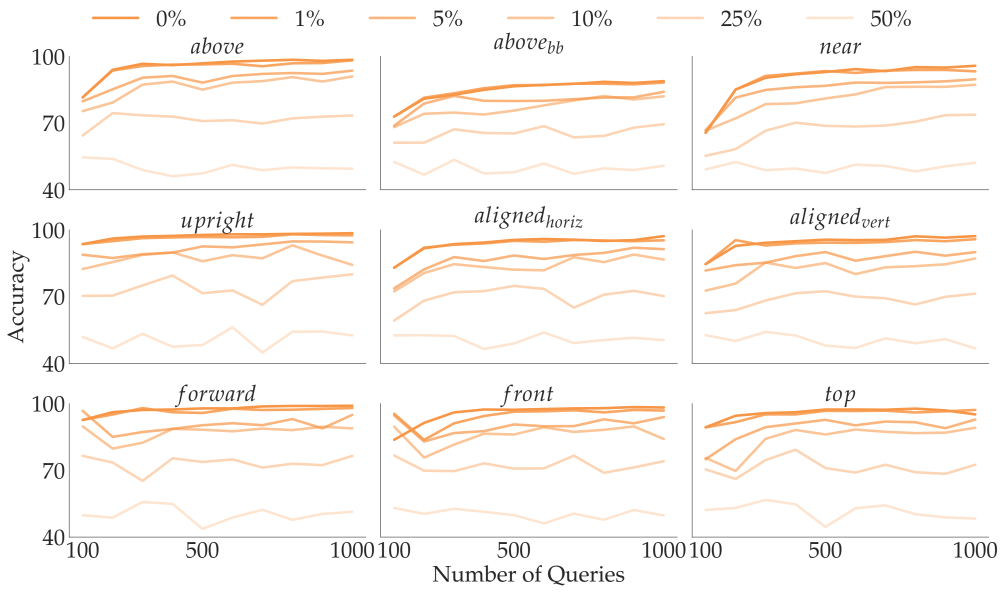


Figure 4.10: Comparison for different labeling noise levels. Our method can withstand reasonable noise levels around 1-10%

and the anchor and moving objects. Given the initial object placement, the robot's goal is to compute a new pose for the moving object that maximizes the feature value with respect to the anchor, grasp the object, and move it to the optimized pose. We used unknown object instance segmentation [294] to segment out the objects, 6-DOF GraspNet [209] to generate grasps, and RRT-connect [166] for motion planning.

We optimize a pose such that when applied to the moving object's (mean-centered) point cloud it results in a scene point cloud that maximizes the feature value. We use the Cross-Entropy Method (CEM) [78] with the cost function given by the feature value of the moved scene point cloud. To encourage the model to find object poses at reasonable orientations, we added a quaternion-angle cost to the CEM optimization, similarly to the metric used in prior work [300]:

$$d(q_1, q_2) = \lambda\,(1 - \langle q_1, q_2 \rangle)$$

where q_1 is the pose being optimized, $q_2 = I = (0, 0, 0, 1)$ is an identity quaternion, and $\lambda = 0.001$ is manually-tuned.

When optimized, our feature models found good poses, even on real-world data of previously unseen objects. Since our method's performance depends on that of the off-the-shelf point-cloud segmentation model used, failures may occur when large portions of the objects are occluded. Moreover, we focus on generating poses that maximize the feature without any collision constraints, so the moved objects sometimes collide with the anchor. In the future, we would fix this by incorporating these features into a planner such as that proposed in [224], so as to include the robot's kinematic constraints directly in the optimization process.

4.6 Discussion

In this chapter, we presented a method for learning relational features with as little expert human interaction is possible. Our approach quickly learns a feature in a low-dimensional space, which is used to generate a large data set for training in a high-dimensional space such as the robot's sensor space.

While our results demonstrate that our features can be used on a 7-DoF Franka Panda arm operating with real sensor data, we still need to investigate how features taught by real people would fare. Our noise analysis in Sec. 4.4.3 suggests that limited random labeling noise might not affect the results too much, but this type of noise might not be a good model for how people make labeling errors. It could also be interesting to study the trade-off between learning accuracy and human burden for different types of queries.

Additionally, while we demonstrated our method in the context of object relations for manipulation, we are excited about future extensions to other types of features: many-object features ("the cup is *surrounded* by plates"), ordering ("sorted from largest to smallest"), or even functional relationships (support / concealment). We could extend PCB to any features where privileged information is available at training time. For instance, if the privileged space contains poses between two frames, we could learn an acceptable speed threshold for manipulating objects. A potential limitation is that the privileged space does become more complex the more (possibly correlated or irrelevant) information we add to it, so learning low-dimensional features may require more data. Our results give some evidence that PCB would overall still require much less data than learning directly from high dimensions, but more future work would be beneficial.

Finally, it could be worthwhile to study modifications to our training and query collection procedure to further improve the quality of the data. For example, we could combine demo and label queries by "warm-starting" the model with demo queries and then actively asking for label queries. The robot could also display examples of the currently learned feature to assist the person in deciding what new examples to give. Lastly, we could consider "chaining" learned features ("mug *upright* and in *front* of the hammer").

Chapter 5

Mapping to a Known Feature Structure

This chapter is based on the paper "Teaching Robots to Span the Space of Functional Expressive Motion" [266], written in collaboration with Arjun Sripathy, Zhongyu Li, Koushil Sreenath, Daniel Brown, and Anca Dragan.

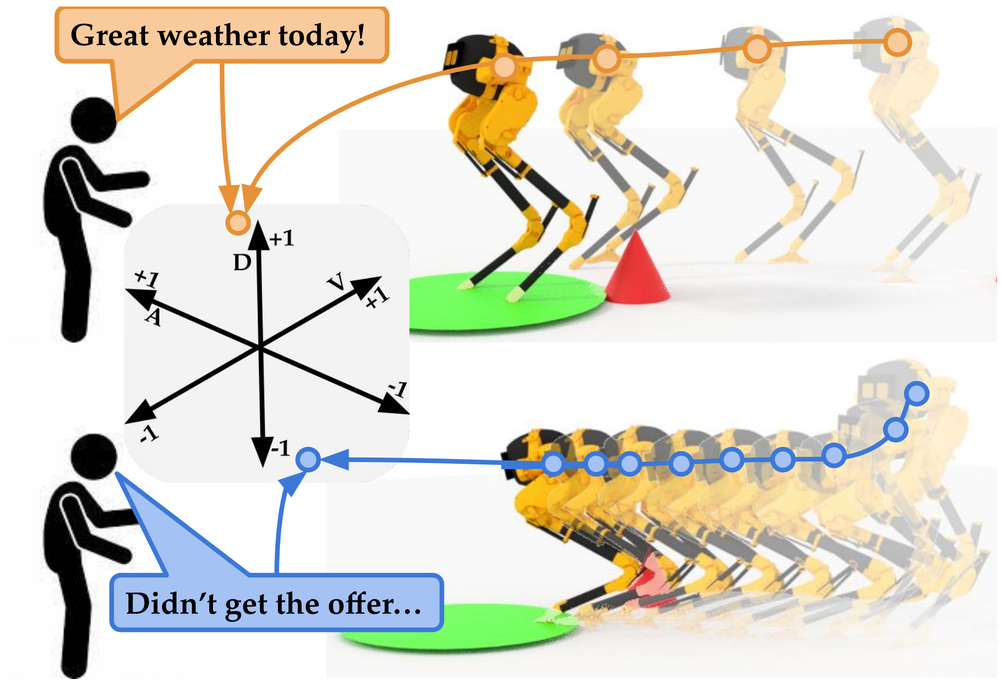


Figure 5.1: Cassie robot performing a task with trajectories it believes exhibit similar emotion VAD as the human utterance. (Top) The person's happier sentiment (orange) results in the robot's fast upright motion. (Bottom) The person's sad sentiment (blue) results in slow and slouched motion.

In Chapters 3 and 4, we assumed that the human can identify the feature representation they wish to teach, but that's not always easy. Imagine you want the robot to understand what comfort means or learn a representation for emotion. How would you even explicate such a feature representation? In this chapter, rather than thinking of and teaching each feature one at a time, we observe that cognitive science grounds certain representations in well-studied *structures*, for which we can then ask the human for labels.

Here we focus on one such structure that centers around representing emotion, and we are specifically motivated by producing robot motion that is expressive. Robotics research tends to focus on generating functional motion, in service of the robot's task. But imagine coming home from work exhausted and disappointed in being rejected from a job application, and the robot continuing to carry on its tasks as if nothing changed. Or, coming back with high energy after taking a walk in the sun, and the robot still putting the dishes away in the same exact way it always does. Ideally, our robots should adapt their behavior like in Fig. 5.1 in response to us, as well as in response to successes and failures they encounter, their confidence levels when performing a task, etc.

While much work has focused on expressive or emotive robot *gestures* [270, 84, 252], the ability to generate emotive *functional* motion that still achieves the robot's task remains an open problem. How can a robot walk to its goal and avoid obstacles *while seeming happy or confident* in response to its user having had a great day? How could a manipulator place a dish in the sink while empathizing with its user's disappointment at work? Taking an existing motion and adjusting its affective aspects, as researchers do with gestures [107, 270], would no longer meet the functional task specification. Instead, prior work [310] has proposed to learn a *cost function* from user feedback for each desired emotion or style, that can be then optimized along with the task specification. Although this addresses the problem of generating motion in an emotive style even as the specifics of the task change, it has the challenge that we have to think of every desired emotion in advance, and collect data specific to it. Further, we still need a way to decide which style or emotion to use.

Here we focus on the fact that emotions are not independent—they are latently related through the Valence-Arousal-Dominance (VAD) spectrum. Motivated by foundational studies in social psychology, VAD identifies three continuous, interpretable features capturing much of emotional variance [56, 246]. In other words, VAD constitutes a structured representation for emotion. Rather than learning independent cost functions for each emotion, our key idea is to learn the mapping from robot trajectories to an emotive VAD latent representation—this way, all user feedback contributes to learning about all emotions, and the robot can model new emotions that interpolate those seen during training. This enables robots to perform tasks in ways expressive of any specific emotion, by optimizing for a trajectory with a projection onto the latent representation that is as close as possible to the desired emotion's VAD. They also may use natural language to infer target emotion VAD: enabling stylistic response to emotive words, like "anger", or even sentences, like "Great weather today!" as in Fig. 5.1.

Our approach interactively collects data from a user to learn this emotive latent space: it starts with an initial space, uses it to optimize emotive trajectories for a variety of task specifications and target emotions, asks the user to label these trajectories, and retrains the latent space to agree with the user labels. Users may choose to label directly with VAD, or use language, which we can map to VAD by using pre-trained language models [85] finetuned to predict VAD scores.

In experiments with simulated human feedback for a Vacuum robot and the Cassie biped, we demonstrate the efficiency of our method in learning emotive costs when compared to approaches which model each emotion independently. We then show in a user study with the Vacuum robot that real humans can teach personalized emotive representations in only 30 minutes of labeling. We find that users are able to recognize target emotions in robot motions generated with the model trained on their labels, even though those target emotions were not explicitly queried for during training. In summary, we propose a method for generating functional, stylistic robot motion by efficiently teaching how trajectories map to the VAD representation, which can then define a cost function encouraging target emotive style specified by natural language. Code and videos are made available at arjunsripathy.github.io/robot_emotive_space.

Despite showing promise in enabling simulated robots to perform functional tasks while expressing a wide range of emotions, much work remains ahead. Demonstrating generalization to a broader range of tasks than locomotion, thinking critically about how target VAD should be determined based on the emotion the user expresses, and moving as much of the process as possible to the physical domain all pose interesting challenges which we discuss further in Sec. 5.4

5.1 Prior Work

Getting robots and virtual avatars to exhibit realistic looking and human-recognizable motions is a well-studied problem, from conveying intent in a task [90, 107, 272], to communicating incapability [168, 273], to expressing emotions [158, 260, 310]. In this section, we focus our attention on literature from the latter category, as our goal is enabling robots to learn emotive styles for performing functional tasks.

Motion style research has its roots in the graphics community. Some work looks at transferring motion capture style from one clip to another [277, 292, 139], but such unconstrained transfer is not appropriate for robots that need to satisfy rigid physical dynamics, or, even more challenging, to still be performing the desired underlying task. Alternative approaches use human demonstrations to learn locomotion styles as cost functions that the robot optimizes to respect task constraints [186, 172]. Unfortunately, due to the correspondence problem in robotics [17], these methods cannot be applied outside of locomotion robots, and acquiring demonstrations of stylized non-anthropomorphic robots is challenging, especially when moving beyond gestures to functional motion.

In typical robotics motion style work, researchers design libraries of emotive motions that the robot can use during task execution [184, 158, 260, 182]. To produce trajectories feasible for complex physical systems, Li et. al. [182] employ a dynamically constrained optimization that encourages the resulting motion to match stylized trajectories while abiding by the robot's dynamics. The motions in these methods are hand-crafted and, therefore, specific to the system and task they are being designed for. To generalize to a more diverse set of tasks, recent methods [310, 186, 172] try to learn a cost function that when optimized produces the desired emotive motion. However, these methods require collecting labels for each emotion one at a time, resulting in inefficient and costly learning that fails to generalize to new emotions.

Instead of representing each emotive motion with an individual cost function, we can learn a more generalizable representation. Suguitan et. al. [270] learned a latent emotive embedding along the VA spectrum [56, 246] capturing a whole space of emotions. While their approach enabled the robot to exhibit simple emotive gestures, like a slow lowering of the head for *sad*, we are interested in integrating emotive motion during the robot's task execution. In this work, we take inspiration to similarly learn an embedding that maps emotive trajectories like the ones in Fig. 5.1 to a latent VAD space but extend their approach to functional task behaviors.

With this embedding, we have a representation that is both generalizable to new emotions and amenable to alternative forms of human feedback, such as natural language. Recently large, pre-trained language models such as BERT [85] have made transfer learning for downstream natural language tasks more accessible and efficient. Further, due to the breadth of research around VAD there exist datasets containing language and corresponding manual VAD annotations [205, 54]. Putting these together we may train a model for mapping natural language into our learned VAD space which will allow us to make the interaction between user and our system even more seamless.

5.2 Approach: A Representation for Expressive Motion

We now present our method for enabling a human to teach robots how to exhibit a broad range of emotions while performing various tasks. The core of our method is training a style discriminator, which predicts what emotion the human would perceive given a trajectory, using VAD labels collected from humans in response to query motions. For any target emotion, we'll define a motion style cost based convincing the discriminator that the trajectory being optimized exhibits the target VAD. We leverage the interpretable structure of VAD as a representation for emotion to improve learning efficiency, interpolate or extrapolate to new emotions, and integrate natural language seamlessly.

5.2.1 Preliminaries

We represent a trajectory $\xi \in \Xi$, where Ξ denotes the set of all trajectories in an environment, as a variable length sequence of waypoints along with the variable time duration between each pair of consecutive points. We denote environment tasks, such as moving from a specific start location to a specific goal, as $\tau \in T$, where T denotes the set of all tasks in the environment. The robot produces a trajectory that solves a particular task in the environment by optimizing a *base cost* $C_{base} : \Xi \times T \to \mathbb{R}$. Optimizing a trajectory using

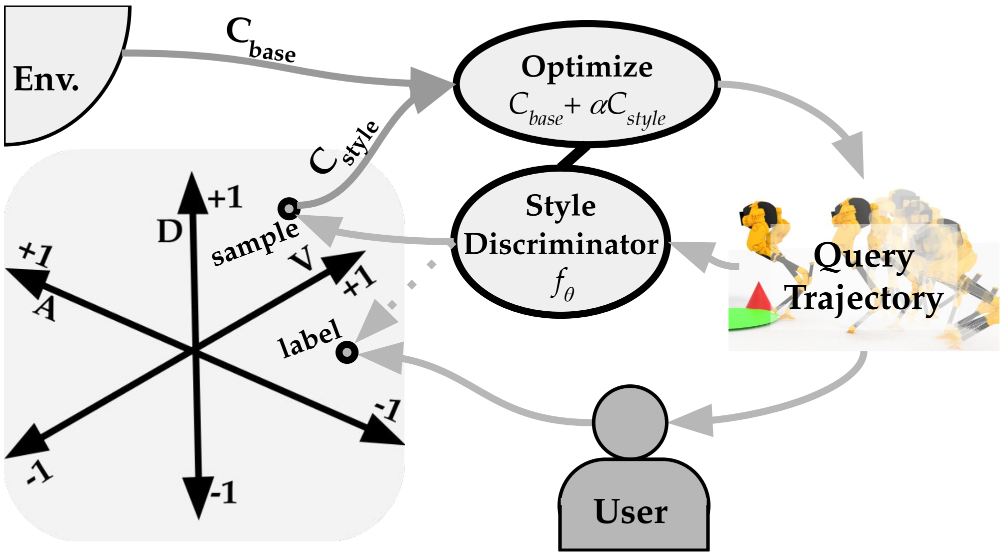


Figure 5.2: Visualization of our method. The optimizer combines the task's base cost C_{base} with the style cost C_{style} of a sampled emotion to produce a query trajectory the style discriminator believes aligns with the target VAD. The user labels the trajectory with a VAD, and the style discriminator f_θ is trained to bring its predicted VAD closer to the human label.

C_{base} yields an efficient trajectory but offers no control over the emotion and typically produces a neutral style.

We describe the style of a trajectory based on an emotion VAD latent $e \in \mathcal{E} := [-1, 1]^3$, where the three values continuously represent Valence, Arousal, and Dominance. Our goal is to learn a trajectory *style cost*, $C_{style} : \Xi \times \mathcal{E} \to \mathbb{R}$, capable of encouraging stylistic alignment with any target emotion e. Ultimately, to produce trajectories that achieve the task with the target style the robot will trade off between the base cost and the style cost:

$$C(\xi, \tau, e) = C_{base}(\xi, \tau) + \alpha \cdot C_{style}(\xi, e) \quad , \tag{5.1}$$

where α is a user specified hyperparameter that prioritizes between style and efficiency.

5.2.2 Cost Function Formulation

To learn the style cost C_{style}, we propose training a neural network style discriminator $f_\theta : \Xi \to \mathcal{E}$ parameterized by θ to map a trajectory ξ to the emotion e the robot exhibits while following it. Our motivation for this design is that every trajectory exhibits some emotion. The style discriminator f_θ naturally motivates a style cost function C_{style} which penalizes a trajectory ξ based on how much its exhibited emotion, $f_\theta(\xi)$, differs from the target e. We formalize this intuition using Euclidean distance in $\mathcal{E}$:

$$C_{style}(\xi, e) = ||f_\theta(\xi) - e||_2^2 \quad . \tag{5.2}$$

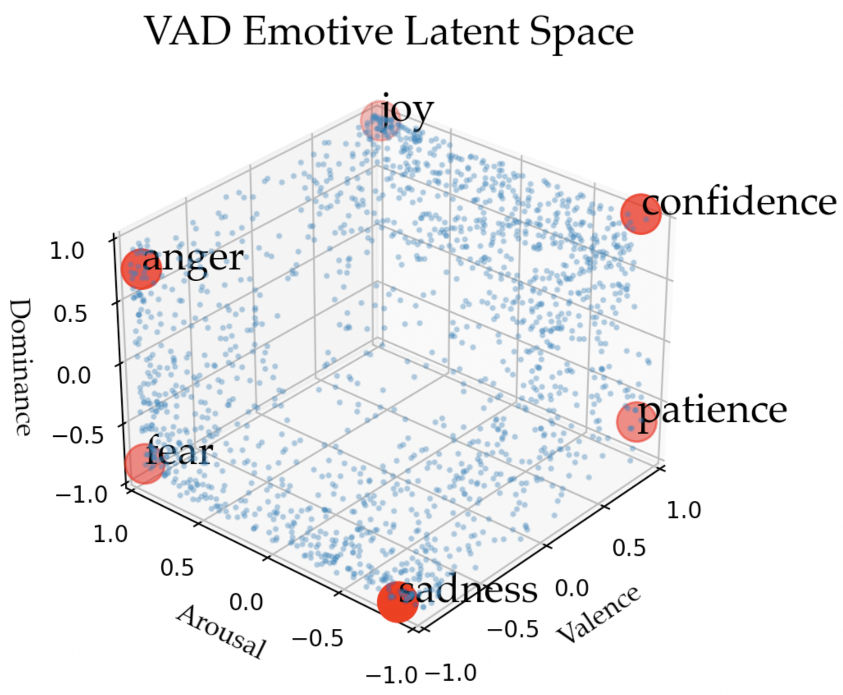


Figure 5.3: The VAD latent space with each of Valence, Arousal, and Dominance being a real valued axis ranging from -1 to 1. The scatter plot depicts the projections of 1,672 emotive words projections into this space with the red dots highlighting the 6 basic evaluation emotions we used in our experiments.

By optimizing the combined cost in (5.1) along with any task constraints, the resulting trajectory ξ completes the task while making its best effort to exhibit the target e.

5.2.3 Generating Human Queries

To train a robust discriminator f_θ, we generate batches of trajectories and query the user for emotive labels as shown in Fig. 5.2. The user provides either direct VAD labels or language which we map to VAD as discussed in Sec. 5.2.5. Our goal is to learn θ, which is randomly initialized and updated after each labeling round as discussed in Sec. 5.2.4.

To generate a round of query trajectories we optimize (5.1) for a batch of sample emotions and tasks, using the current estimate of θ for C_{style}. Given C_{style} does not explicitly model the task, we focus on how to sample emotions in a way that is most informative for θ. Motivated by active learning literature we seek a diverse batch that biases towards *important*, unexplored areas of $\mathcal{E}$ [308]. Emotions are not uniformly spread across $\mathcal{E}$, and we would like to focus our queries on more populated areas of the space. We leverage the empirical emotion distribution from the NRC VAD lexicon [205], which contains annotated VAD values for 20k words. We filtered them down to 1,672 common emotive words, resulting in the VAD distribution in Fig. 5.3.

We now propose an active learning method for improving query coverage of this distribution to make the discriminator, f_θ, more robust. For the first round of queries, since the network f_θ is randomly initialized and has no semantic meaning yet, we uniformly sample B emotions from $\mathcal{E}$. To explain the process for successive rounds we must establish some notation. We conduct K query rounds with $k \in [1, K]$ referencing the round index. Let $\mathcal{S}_k := \{s_k\}^{1:B} \in \mathcal{E}^B$ reference the batch of sample emotions to be chosen by active learning to cover the empirical distribution of $D = 1672$ lexicon VAD values $e_1, ..., e_D$. By optimizing (5.1) for $\mathcal{S}_k$ alongside tasks randomly presented by environment, we generate query trajectories $\mathcal{Q}_k := \{q_k\}^{1:B} \in \Xi^B$. For these queries we will collect human labels referenced $\mathcal{L}_k := \{l_k\}^{1:B} \in \mathcal{E}^B$.

Our active learning method seeks to minimize the average distance between lexicon emotions and the closest acquired label from any round. This relies on estimating the $\mathcal{L}_k$ based on our selection of $\mathcal{S}_k$. Our approximation here is $\mathcal{L}_k \approx \mathcal{S}_k$ which becomes more accurate over the course of training. With this assumption, for $k > 1$, we may select $\mathcal{S}_k$ as:

$$\mathcal{S}_k = \arg\min_{\mathcal{S}_k} \sum_{i=1}^{D} \min_{l \in \mathcal{S}_k \cup \bigcup_{j=1}^{k-1} \mathcal{L}_j} ||e_i - l||^2 \ . \tag{5.3}$$

$\mathcal{S}_k$ will bias towards dense areas of the VAD space where we do not yet have a nearby label. We approximate the optimal solution using expectation maximization [81]. We next look at how to update θ after each round based on the collected feedback.

5.2.4 Trajectory Network Training and Architecture

After each round k of querying, we update θ given our trajectory queries and VAD label responses collected so far ($\mathcal{Q}_{1:k}$, $\mathcal{L}_{1:k}$). We optimize the following MSE training loss:

$$L_k(\theta, \mathcal{Q}_{1:k}, \mathcal{L}_{1:k}) = \sum_{i=1}^{k} \sum_{j=1}^{B} ||f_\theta(q_i^j) - l_i^j||_2^2 \ . \tag{5.4}$$

Note the summand is exactly equivalent to $C_{style}(q_i^j, l_i^j)$ allowing for an alternative interpretation: we treat the queries as demonstrations for the emotion labels and would like to assign them minimal style cost.

In implementing this method, we have to choose a specific architecture for f_θ. Recall from Sec. 5.2.1 that we represent a trajectory as a variable-length sequence of waypoints and time deltas. We utilize an architecture similar to PointNet [227] for its simplicity and ability to gracefully handle varying length trajectories. First, a fully connected network processes each waypoint independently. Then we apply average and softmax pooling over waypoints to produce a single trajectory embedding. From there another fully connected network predicts the overall trajectory VAD value. Both networks use ELU activation [72].

We want network predictions to be smooth so they guide trajectory optimization well when used within the cost function. In other words, not only must predictions be accurate, but their gradient signal must also be informative. These factors motivated us to use smoother pooling (average & softmax) and activation functions (ELU), and to limit network capacity. A single hidden layer in each network, of dimensions chosen to match the complexity of the robot, along with L1 regularization worked well in our experiments.

5.2.5 Natural Language to VAD

We now describe how VAD may be inferred from natural language and where this may be used. We look up single words directly in the NRC VAD lexicon [205]. For sentences or words not present in the lexicon, we apply a BERT model finetuned to predict VAD using EmoBank: a dataset with 10k VAD labeled sentences [85, 54]. The wealth of resources and data around VAD is another benefit of using the spectrum as our latent space.

In many scenarios language provides a more natural means of communicating emotion than VAD. During training, language labels could be easier to provide compared to VAD directly. After training, the robot should determine target emotion in a less burdensome way than explicitly requesting VAD. Interpreting VAD from language allows the robot to seamlessly identify target emotion and modulate its behavior around humans.

5.3 Experiments: Expressive and Functional Robot Motion

There are three primary hypotheses we seek to test with our experiments. (1) A real human is capable of using our method to teach their perception of a robot's emotive style, and after training they perceive the robot's intended target emotion in generated stylistic trajectories; (2) Our method is more efficient at learning a set of emotive styles than an approach that models each emotion independently; (3) Despite using general emotive labels, our method is equally efficient at learning any single emotive style as alternatives which leverage feedback specific to that emotion.

To evaluate hypothesis (2) & (3) we run a set of simulated human trials comparing the query efficiency our method to alternatives which model emotions independently. To test (1) we conduct a user study where real humans' perception of emotion takes the place of simulated human heuristics. We evaluate the effectiveness of teaching by the extent to which the human (simulated or real) perceives emotion similar to what the robot intended to exhibit while completing various tasks. We will discuss the results for each case and ultimately find our hypotheses supported by the data collected.

5.3.1 Robots

We used two simulated robots to test our method: a simpler Vacuum robot (VacuumBot) and the more complex Cassie bipedal robot [182]. We now describe the robot

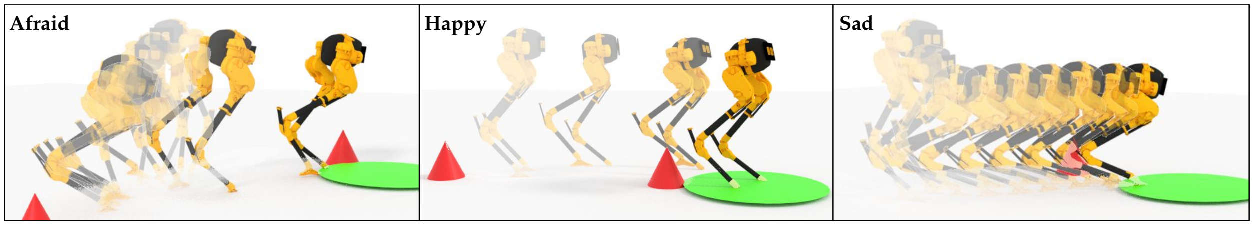


Figure 5.4: Cassie must reach a target location avoiding obstacles represented by cones. Here it optimizes for various target emotions on various tasks using a model learned with our method. Visuals overlay one snapshot per second with earlier frames made more transparent. *Afraid* first takes a few cautious steps back before proceeding. *Happy* keeps head high and moves fast. *Sad* slouches and proceeds slowly.

specifications, the environments they operated in, and the tasks they must complete.

VacuumBot is tasked with collecting dust that appears in a 2D world. It has 3 DOF controlling horizontal, vertical and angular acceleration and is subject to various physical constraints including gravity and friction. The current state of the robot and environment is summarized by the position and velocity corresponding to the DOF, and the location of the dust. Trajectories are optimized for VacuumBot entirely using PyTorch [221].

Cassie, shown in Fig. 5.4, is a person-sized bipedal robot which has 20 DoF including 6 DoF of the base (its pelvis), 5 DoF and 2 passive joints of each leg. More details about Cassie can be found in [182]. Cassie, shown in Fig. 5.4, is a person-sized bipedal robot which has 20 DoF including 6 DoF of the base (its pelvis), 5 DoF and 2 passive joints of each leg. More details about Cassie can be found in [182]. In our paper, Cassie is tasked with navigating to a randomly generated target location while avoiding random obstacles represented by the red cones in Fig. 5.4. We leverage collocation to obtain a trajectory minimizing the proposed style cost while imposing constraints including collision-avoidance and reduced-order nonlinear dynamics as described in [183]. Based on the optimized trajectory of robot base velocity and height, the robot's whole body motion is obtained from a gait library optimized by its full-order dynamics [182, 132]. This nonlinear optimization is formulated in CasADI [16], solved via IPOPT [34], and the resulting trajectory is visualized through animation in Blender [182]. To integrate our PyTorch implementation of f_θ we export our learned θ after every training round and replicate the neural network as a fixed numerical function in CasADI [16].

5.3.2 Experimental Design

The simulated human trials and real human study used a very similar experimental design. In this section, we describe the process for evaluating a single method in general terms, and will discuss how this was adapted for the two contexts in their corresponding sections. We implement alternative models in a way that allows them to conform to the same evaluation procedure as our method.

Using notation from Sec. 5.2.4, we train by conducting K rounds of B trajectory query batches updating the model after each batch of labels. Then we evaluate stylistic trajectories, produced for a representative set of evaluation emotions, based on the extent to which the human perceives the intended emotion in each.

To identify our evaluation emotions, we again leveraged VAD values from the NRC VAD Lexicon [205]. Based on the empirical distribution shown in Fig. 5.3, we identified the corners [-1, -1, -1], [1, 1, 1], [-1, 1, -1], [1, -1, 1], and [-1, 1, 1] as the best regions to evaluate our model due to their population density and general coverage of the space. We selected representative emotions near each of these corners: *sadness*, *joy*, *fear*, *confidence*, *anger*, and *patience* respectively. Note that consecutive pairs on this list are diametric opposites in VAD space. We will not always use all 6 evaluation emotions and use N to reference the specific number we are working with. For consistency, when $N < 6$ we will always use the *first* N emotions based on the order we presented these above. We restrict ourselves to $N \in \{2, 4, 6\}$ to keep the emotions in opposing pairs.

Our three evaluation metrics are *Quality* score, *Top-1* accuracy, and *Top-2* accuracy. Quality score measures binary alignment: how well trajectories express the intended emotion compared to its diametric opposite. Top-X accuracies measure precise alignment: how well trajectories express the intended emotion compared to all $N - 1$ alternatives. Ideally we'd evaluate across all tasks, but given the task space is continuous we randomly sample M tasks for each of the N evaluation emotions averaging the metric values we get.

To compute the quality score, the robot presents the user with $N/2$ sets of $2 \cdot M$ trajectories for evaluation. Each set is associated with one diametric pair of evaluation emotions, say A and B. The $2 \cdot M$ trajectories includes M trajectories optimized for A and M trajectories optimized for B. The user is asked to assign each trajectory a score, s from 1 to 7 answering the Likert question: *Is the trajectory more expressive of Emotion B than A?* with a response of 1 indicating the trajectory is very expressive of Emotion A and a 7 indicating that the trajectory is very expressive of B. Let q be the Quality score metric. For trajectories optimized for B we define $q \coloneqq s$, and for A $q \coloneqq 8 - s$. As a result, q ranges from 1 to 7 with 7 indicating perfect alignment with the intended emotion compared to its opposite.

To compute Top-X accuracies, we present the user with another $N \cdot M$ trajectories including M trajectories optimized for each of the evaluation emotions. Now the user is asked to select which of the N emotions is most expressed by each trajectory as well as their second choice. We define the Top-X accuracy metric, for $X \in \{1, 2\}$, as the proportion of the time the user's top X choices include the intended emotion.

5.3.3 Simulated Human Trials

We first conducted a set of experiments with simulated human feedback, since it would have been impractical to reliably test all our configurations with real humans.

5.3.3.1 Simulating Human Feedback. The simulated "human" (SH) uses heuristics to determine VAD for trajectories. For example, SH quantifies dominance for Cassie based on

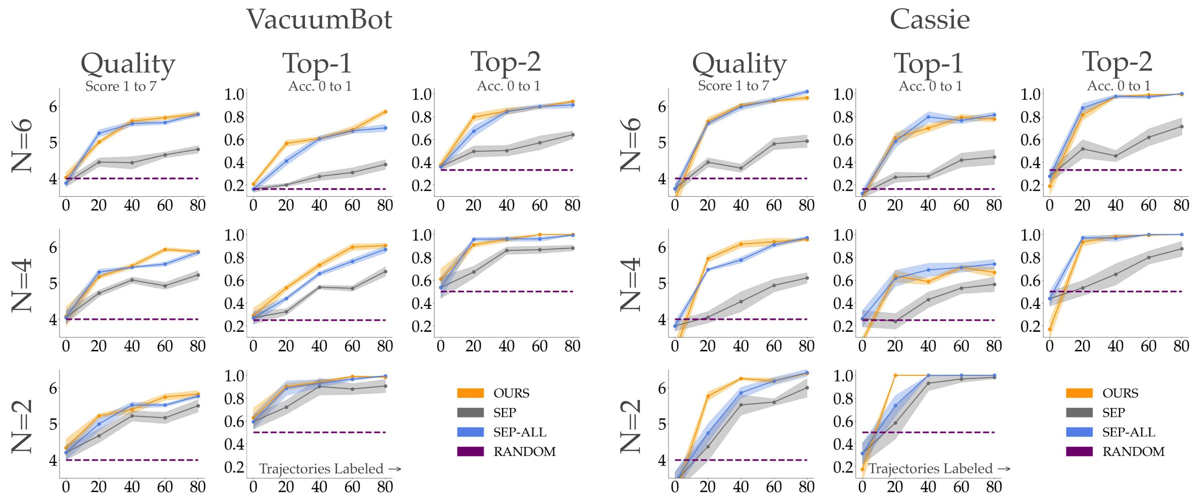


Figure 5.5: Quality score, Top-1, Top-2 accuracy and standard errors during training averaged across six seeds for each (environment, method, N) configuration. Metrics were computed before training and after each of 4 batches of 20 trajectory queries, with the query number indicated in the horizontal axis. Shown in dotted lines are the expected values if users chose randomly during evaluation. Our method matches performance of SEP-ALL whereas SEP struggles to keep pace as the number of emotions N increases.

the average head height. SH may not accurately represent human emotive perception, but its consistent feedback allows us to compare learning efficiency between various methods.

During training SH directly gives its VAD for a trajectory as feedback. During evaluation, it must further transform this VAD value to mimic appropriate human responses. For the Likert question juxtaposing opposite emotions, SH projects the VAD value on to the diametric axis between the pair of emotions; then SH linearly scales the result so Emotion A is 1 and B is 7. For the choice based component, SH picks the closest and second closest evaluation emotion based on Euclidean distance in VAD space.

5.3.3.2 Alternative Methods. We compare our method to two approaches that model emotions independently. First is an approach inspired by Zhou et al. [310] which we'll reference as SEP because it trains *separate* models directly predicting cost for each evaluation emotion. SEP directly asks the user to label trajectories with how expressive they are of *one* of the evaluation emotions. The second approach is SEP-ALL which we allow access to real valued cost labels for *all* evaluation emotions for each trajectory; SEP-ALL does not have to split its labeling budget between emotions as SEP does. Recall our approach requests VAD labels from the user irrespective of the evaluation set enabling generalization beyond predefined emotions. In contrast, SEP and SEP-ALL both require knowledge of the evaluation set of emotions prior to training and get feedback specific to them.

To select emotions to generate trajectory queries for, SEP and SEP-ALL simply sample with replacement from the evaluation set. To remove a potential confound and isolate

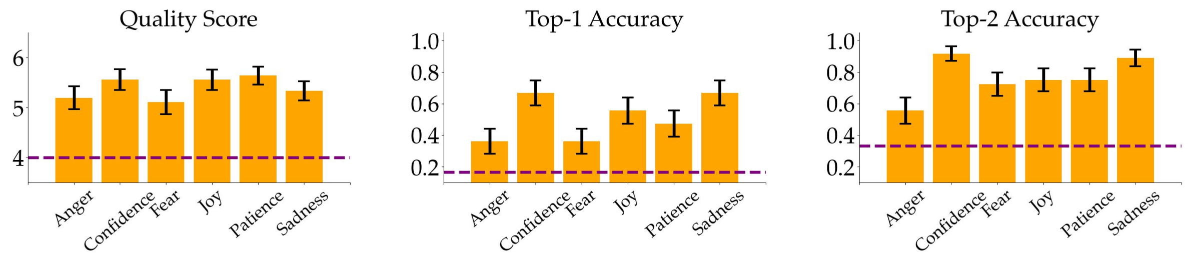


Figure 5.6: Quality score, Top-1, and Top-2 accuracy and standard error for each evaluation emotion averaged across study users. Metrics significantly outperforms a random guess baseline, shown with the dashed lines, suggesting humans can indeed teach emotive motion with our method.

learning efficiency, we use the same selection process for our method in the simulated experiments. In the Sec. 5.3.4 study we revert to the active learning described in Sec. 5.2.3.

5.3.3.3 Simulation Results Discussion. The simulated human trials involved running each (environment, method, N) combination with 6 seeds, 108 experiments total, using $K = 4$ rounds of $B = 20$ trajectory labels. We present the average evaluation results using $M = 6$ tasks per emotion along with standard errors in Fig. 5.5.

To evaluate hypothesis (2), improving learning efficiency for a set of emotions compared to an approach that models them independently, we juxtapose our method with SEP. Across all metrics our method is able to reach a higher performance faster, the gap growing with N, supporting our hypothesis. With SEP each query is only informative for one of N emotive models. By contrast, with our method each query is informative for the entire VAD space and, thus, every evaluation emotion to some extent.

To evaluate hypothesis (3), matching learning efficiency for a single emotion compared to an approach that gets feedback specific to that emotion for each query, we juxtapose our method with SEP-ALL. SEP-ALL gets emotion specific cost labels for all N emotions with each query, as opposed to the generic VAD label our method receives. Yet across the board performance of our method matches SEP-ALL supporting our hypothesis again. It is not practical to go beyond a few evaluation emotions with SEP-ALL since labeling overhead scales linearly with respect to N, whereas our method has constant overhead enabling capture of the full span of emotions. Furthermore, even for small N providing VAD values (or natural language) may be easier than real valued emotion specific costs.

Ultimately, VAD provides an interpretable latent representation that allows efficient learning of the space of emotive style with performance no worse than if we targeted any specific target emotion. Fig. 5.4 visualizes some emotive styles Cassie learned from this experiment, demonstrating our method's ability to work with high DOF, complex robots.

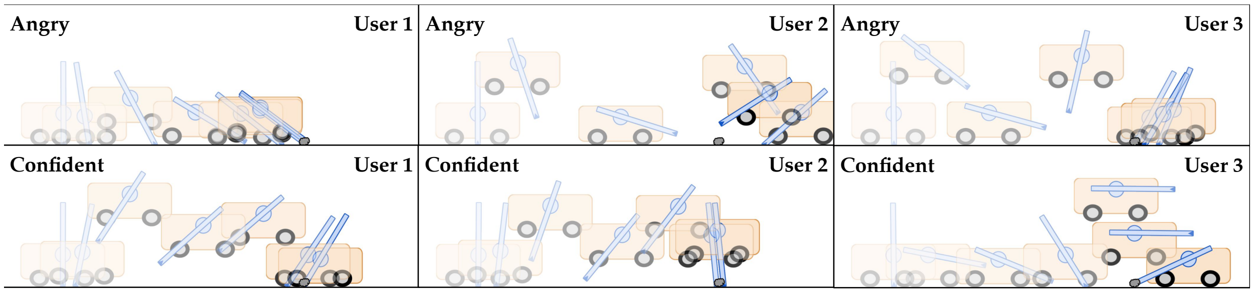


Figure 5.7: VacuumBot collecting dust with style trained by three real users optimized for two target emotions on a single task. Judging by these motions, User 2 and 3's perception of anger involved greater speed, jumping, and arm movement than User 1. User 3's confidence had more arm movement but less jumping than User 1 and 2's.

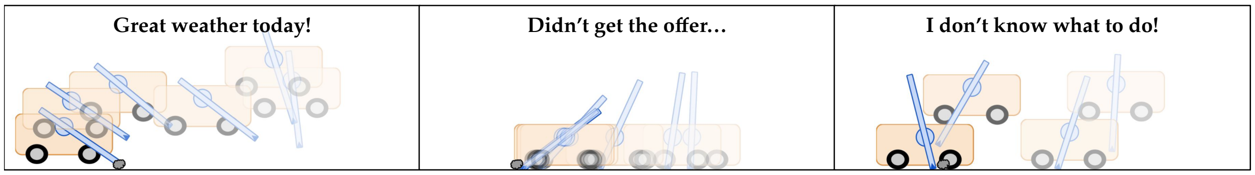


Figure 5.8: Three trajectories for a single task where VacuumBot infers VAD from the displayed phrases and produces motions based on a study user's trained model. When the human expresses cheerfulness it gracefully hops to the goal. When the human expresses sorrow it slowly slouches its way there. When the human expresses fear it reflects that nervous energy.

5.3.4 Real Human Study

The efficiency demonstrated in Sec. 5.3.3 shows that real humans may feasibly use our system. We present a user study with VacuumBot aimed at testing hypothesis (1): the effectiveness of our method in teaching emotive style that is recognizable to end users.

5.3.4.1 Study Setup. We recruited 12 participants (9 male, 3 female) aged 20-27. They were asked to provide emotion labels for $K = 2$ rounds of $B = 20$ robot trajectories. We use a lower value for K here compared to the simulated experiments to emphasize the practicality of our approach. We found it was easier for humans to consistently label trajectories with VAD labels; however, as discussed earlier language may be used as well.

For the evaluation phase we used $N = 6$ emotions with $M = 3$ tasks each for all participants. To keep the overall study time shorter we did not perform intermediate evaluations, only evaluating after all labeling was complete. The labeling portion of the study took 30-40 minutes and the evaluation phase 20-25 minutes per participant.

5.3.4.2 Study Results Discussion. Fig. 5.6 compares human evaluation results to a random guess baseline. We break down results by each of the 6 evaluation emotions.

For every emotion and metric we ran t-tests at the 5% significance level comparing performance to the random baseline. Each test indicated statistically significant improvement which is reflected by the standard errors in Fig. 5.6. These results support hypothesis (1): our participants could teach the robot by labeling query trajectories in about 30 minutes, and during evaluation they perceived the robot's intended emotion at a rate significantly higher than random chance.

While the quality scores are relatively consistent across emotions the Top-1 accuracy varies a fair amount. This suggests the method is reliable in producing trajectories that are generally in the right direction but might not exactly line up with the intended one. In some cases the former may be of primary importance and in others emotional precision may be equally important; however, it is reassuring to know that the robot will roughly align with the intended style even when it is not perfect in targeting the particular one.

A qualitative observation is that all users had their own personalized views of emotions. As demonstrated in Fig. 5.7, robots trained by three different study users ended up with fairly different, yet justifiable behaviors for the same emotions and task. Furthermore, in Fig. 5.8 we showcase example motions for one user's robot generated based on the VAD of short phrases. This highlights how our method enables the robot to learn more than a finite set of emotions. It learns an entire emotive space which it may index into to generate appropriately expressive behavior. By making the teaching process more accessible, our work takes an important step towards enabling anyone to teach robots nuanced behaviors without needing a technical foundation themselves.

5.4 Discussion

Summary. We introduced a method that enables robots to perform functional tasks in ways that are expressive of a wide range of emotions. After being taught how trajectories map to VAD the robot may include a cost function encouraging a target emotive style in task motion optimization. Natural language VAD inference enables the robot to decide target emotions while in use and may also substitute numerical VAD labels during training. Our experiments suggest that learning the VAD space jointly, beyond enabling emotion generalization, is more efficient and practical than trying to model each target emotion separately. Furthermore, our experiments provide evidence that our method enables real humans to teach robots discernible, emotive style.

Limitations and Future Work. First we share some short term directions. our environments only presented robot locomotion tasks, albeit varying the start, goal, and obstacle locations, hence expanding the task space to include more diverse objectives (e.g. object manipulation) would be an interesting direction. Although we mentioned the possibility of using language in place of VAD for training we did not explicitly evaluate this option. After training we propose inferring target VAD from user sentences, and despite promising qualitative results in Fig. 5.8 more in depth analysis is required. It's unclear even whether the robot should alter user emotion or merely reflect it.

Now we highlight some long term challenges. There are existing solutions for producing physical trajectories with our style cost [182], but bringing the training procedure into the physical domain is more challenging. It removes the ability to easily reset the robot as we do in simulation to facilitate label collection. Another challenge is while VAD captures the three most important emotive directions, sometimes differing emotions have similar VAD. For example, *fear* and *disgust* both have low valence and dominance with high arousal. Future work would have to navigate these subtleties while preserving the efficiency of our learning process.

We are excited about our results and believe they make an important contribution towards the end-goal of making robots more expressive and enabling people to teach personalized emotive styles. We look forward to seeing robots operate alongside humans with control over their exhibited emotion.

Chapter 6

All-at-Once Representation Learning

This chapter is based on the paper "SIRL: Similarity-based Implicit Representation Learning" [45], written in collaboration with Yi Liu, Rohin Shah, Daniel Brown, and Anca Dragan.

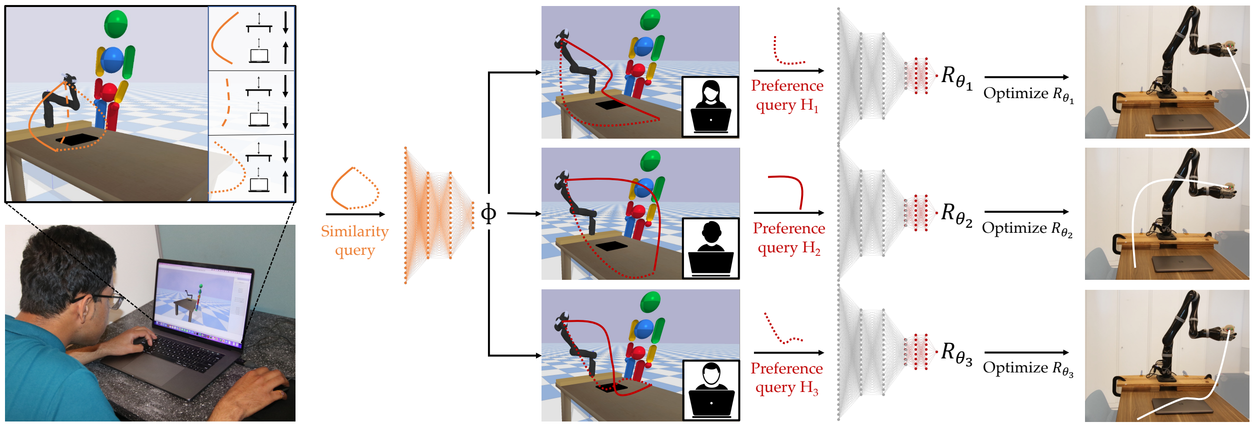


Figure 6.1: Our goal is to learn representations for robot behavior that capture what is salient to people, and, thus, support generalizable preference learning with low sample complexity. We propose to extract this representation by asking people trajectory similarity queries (left), where they judge which two out of three trajectories are most similar to each other. We then use the representation to learn reward functions corresponding to different people's preferences on different tasks (right).

When the human teaches their representation explicitly, they have to either know how to break it down into the features that compose it (like in Chapters 3 and 4) or assume a well-studied structured representation (like in Chapter 5). While this expectation is oftentimes reasonable, people may not always be able to explicate their representation and break it down into concepts that are individually teachable. In this chapter, we study how to enable the robot to also *implicitly* extract the person's representation by having them solve *representation-specific tasks* – proxy tasks designed to learn an embedding of what matters to the human all-at-once from their behavior.

Just like in Chapter 3, we are motivated here by learning representations that lead to learning generalizable rewards. For instance, imagine waking up in the morning and your home robot assistant wants to place a steaming mug of fresh coffee on the table exactly where it knows you will sit. Depending on the context, you will have a different preference for how the robot should be doing its task. Some days it carries your favorite mug close to the table to prevent it from breaking in the case of a slip (so that it will remain your favorite mug); other days the steam from your delicious meal is difficult to handle for the robot's perception, so you'd want it to keep a large clearance from the table to avoid collisions. Similarly, some days you want the robot to keep your mug away from your laptop to avoid spilling on it; other days the mug only has an espresso shot so you want the robot keep it close to the laptop to prevent clutter and leave the rest of the table open for you.

The reward function the robot should optimize changes — whether due to variations in the task, having different users, or, as in the examples above, different contexts that are not always part of the robot state (e.g. holding the user's favorite mug and not just a regular mug). However, the *representation* on top of which the reward is built, i.e the *features* that are important (like the distance from the table, being above the laptop, etc.), are shared. If the robot learns this representation correctly, it can use it to obtain the right reward function, even if the task, user, and context changes. Meta-learning and multi-task learning methods [296, 109, 215] learn the representation from user input meant to teach the full reward, like preference queries or demonstrations. By contrast, we propose that if learning generalizable representations is the goal, then we should ask the user for input that is specifically meant to teach the representation itself, rather than input meant to teach the full reward and hoping to extract a good representation along the way.

But asking people to teach robots representations, rather than tasks, is not so easy. What *are* the features that they care about? While some techniques advocate for people enabling users to teach each feature separately [40], people may not always be able to explicate their representation and break it down into concepts that are individually teachable. In this chapter, our idea is that we can implicitly tune into the representations people use by asking them to do a proxy task of evaluating *similarity* of behaviors. Behaviors will be similar if the features that matter are similar, even if low-level behavior is different; conversely, behaviors will be different if even one of the features that matter differs. This, in turn, should enable the robot to arrive at the features that matter — we want robots that can disambiguate between what needs to go into the representation versus what is spurious, as well as what aspects of behavior can be compressed together into a feature embedding versus kept separate. We thus introduce a novel type of human input to help the robot extract the person's representation: *trajectory similarity queries*. A trajectory similarity query is a triplet of trajectories that the person answers by picking the two more similar trajectories. In Fig. 6.1 (left), the person chooses the two trajectories that are close to the table and far from the laptop, even though visually they look dissimilar. This results in an (anchor, positive, negative) triplet that can be used for training a feature representation. We call this process Similarity-based Implicit Representation Learning (SIRL).

Our method has a parallel in self-supervised learning work, especially contrastive learning, where the goal is to learn a good visual representation by training from (anchor, positive, negative) triplets generated via data augmentation techniques [65]. However, this notion of similarity is purely visual, driven by manually designed heuristics for data augmentation, and is not necessarily reflective of what users would consider similar. For instance, two images might be labeled as visually different, when in fact their difference is only with respect to some low-level aspects that are not really relevant to the distribution of tasks people care about. This would result in representations that contain too many distractor features that are not present in the human's representation. Our method uses similarity too, but we defer to the user's judgement of similarity, with the goal of reconstructing the *user's representation*.

Of course, our method is not the full answer to learning causally aligned representations. But our experiments suggest that it outperforms methods that are self-supervised, or that learn from input meant to teach the full tasks. In simulation, where we know the causal features, we show that SIRL learns representations better aligned with them, which in turn leads to learning multiple more generalizable reward functions downstream (Fig. 6.1). We also present a user study where we crowdsource similarity queries from different people to learn a shared SIRL representation that better recovers each of their individual preferences. While the study results do show a significant effect, the effect size is much lower than in simulation. This is attributable in part to the interface difficulty of analyzing the robot trajectories, which means more work is needed to determine the best interfaces that enable users to accurately answer similarity queries. Moreover, some users reported struggling to trade off the different features, which means that similarity queries might not be entirely preference-agnostic. Nonetheless, our results underscore that there are gains by explicitly aligning robot and human representations, rather than hoping it will happen as a byproduct of learning rewards from standard queries.

6.1 Prior Work

Learning from Human Input. Human-in-the-loop learning is a well-established paradigm where the robot uses human input to infer a policy or reward function capturing the desired behavior. In imitation learning, the robot learns a policy that essentially copies human demonstrations [219], a strategy that typically doesn't generalize well outside the training regime [179]. Meanwhile, inverse reinforcement learning (IRL) uses the demonstrations to extract a reward function capturing *why* a specific behavior is desirable, thus better generalizing to unseen scenarios [2]. Recent research goes beyond demonstrations, utilizing other types of human input for reward learning, such as corrections [25], comparisons [290], or rankings [49]. Unless explicitly designed for, these methods learn a latent representation implicitly from the respective human input. We seek to instead explicitly learn a preference-agnostic latent space that can be used for downstream reward learning. We focus on learning reward functions via preference queries [290], but we believe the latent space we learn can be useful for learning from any of the above types of feedback.

Representation and Similarity Learning. Common representation learning approaches are unsupervised [66, 133, 64] or self-supervised [87, 222, 22, 170], but because they are purposefully designed to bypass human supervision, the learned embedding does not necessarily correspond to features the person cares about. Prior work leverages task labels [62] or trajectory rankings [50] to learn latent spaces for specific goals or preferences. By contrast, we focus on learning task-agnositic measures of feature similarity that are useful for learning multiple preferences. Some work looks at having people interactively select features from a pre-defined set [58, 55, 196] or teach task-agnostic features sequentially via kinesthetic feature demonstrations [41] or active learning techniques [167, 129, 42]. We instead focus on fully learning a lower-dimensional feature representation all-at-once, rather than one at a time. Furthermore, rather than relying on the human to provide physical demonstrations for learning a good feature space [40, 41], we propose a more accessible and general form of human feedback: showing the user triplets of trajectories and simply asking them to label which two trajectories are the most similar. Triplet losses have been widely used to learn similarity models that capture how humans perceive objects [6, 274, 203, 82, 12]; however, to the best of our knowledge, we are the first to use a triplet loss to learn a general, task-agnostic similarity model of how humans perceive trajectories.

Meta- and Multi-Task Reward Learning. To learn multiple reward function models, prior work has proposed clustering demonstrations and learning a different reward function for each cluster [86, 23, 70]; however, these methods require a large number of demonstrations and do not adapt to new reward functions. Meta-learning [95] seeks to learn a reward function initialization that enables fast fine-tuning at test time [297, 302, 141, 257]. Multi-task reward learning approaches pretrain a reward function on multiple human intents and then fine-tune the reward function at test time [109, 215]. This has been shown to be more stable and scalable than meta-learning approaches [201], but still requires curating a large set of training environments. By contrast, we do not assume any knowledge of the test-time task distribution *a priori* and do not require access to a population of different reward functions during training. Rather, we focus on learning a task-agnostic feature representation that can be utilized for down-stream reward learning tasks. In particular, we test our learned representation on the down-stream task of learning models of human reward functions via pairwise preference queries over trajectories [290, 251, 36, 180].

6.2 Approach: A Proxy Task for Representation Learning

We present our method for learning preference-agnostic representations from trajectory similarity queries. Our intuition is that if a human judges two behaviors to be similar, then their representations should also be similar. Since directly asking if two trajectories are similar is difficult without an explicit threshold, we instead present the human with a triplet of trajectories and ask them to pick the two most similar (or, equivalently, the most dissimilar one). We use the human's answers to train the representation such that similar trajectories have embeddings that are close and dissimilar trajectories map to embeddings far apart. The robot then uses this latent space as a shared representation for downstream preference learning tasks with multiple people, each with different preferences.

6.2.1 Preliminaries

We define a trajectory ξ as a sequence of states, and denote the space of all possible trajectories by Ξ. The human's preference over trajectories is given by a reward function $R : \Xi \mapsto \mathbb{R}$ that is unobserved by the robot and must be learned from human interaction. The robot reasons over a parameterized approximation of the reward function R_θ, where θ represents the parameters of a neural network. To learn θ, the robot collects human preference labels over trajectories [290, 71] and seeks to find parameters θ that maximize the likelihood of the human input. The robot can then use the learned reward function to score trajectories during motion planning in order to align its behavior with a particular human's preferences. We focus on explicitly using human input to first learn a good representation and then use that representation for downstream reward learning, rather than using reward-specific human input (e.g., preferences or demonstrations) to implicitly learn the representation at the same time as the reward function.

6.2.2 Training the Representation via Trajectory Similarity Queries

We seek to train a latent space that is useful for multiple downstream preference learning tasks. To do this, we propose learning a preference-agnostic model of human similarity. One way to learn such a model would be to ask users to judge whether two trajectories are similar or not; however, humans are better at giving relative rather than binary or quantitative assessments of similarity [155, 268]. Thus, rather than asking users to use some internal threshold or scoring mechanism to quantitatively measure similarity, we instead focus on qualitative trajectory similarity queries. We present the user with a visualization of three trajectories and ask them to pick the two most similar ones (equivalently the most dissimilar one). The human's queries form a data set $\mathcal{D}_{sim} = \{(\xi^i_{P_1}, \xi^i_{P_2}, \xi^i_N)\}$, where $\xi^i_{P_1}$ and $\xi^i_{P_2}$ are the trajectories that are most similar and ξ^i_N is the trajectory most dissimilar to the other two.

We can interpret similarity (or dissimilarity) as a distance function, so we define the distance between two trajectories as the L_2 feature distance: $d(\xi_1, \xi_2) = \|\phi(\xi_1) - \phi(\xi_2)\|_2^2$. Given a dataset of trajectory similarity queries $\mathcal{D}_{sim}$, we use the triplet loss [28]:

$$\mathcal{L}_{trip}(\xi_A, \xi_P, \xi_N) = \max(d(\xi_A, \xi_P) - d(\xi_A, \xi_N) + \alpha, 0) \ , \tag{6.1}$$

a form of contrastive learning where ξ_A is the anchor, ξ_P is the positive example, ξ_N is the negative example, and $\alpha \geq 0$ is a margin between positive and negative pairs. However, because our queries do not contain an explicit anchor, our final loss is as follows:

$$\mathcal{L}_{sim}(\phi) = \sum_{i=1}^{|\mathcal{D}_{sim}|} \mathcal{L}_{trip}(\xi^i_{P_1}, \xi^i_{P_2}, \xi^i_N) + \mathcal{L}_{trip}(\xi^i_{P_2}, \xi^i_{P_1}, \xi^i_N) \ . \tag{6.2}$$

We train a similarity embedding $\phi : \Xi \mapsto \mathbb{R}^d$ that minimizes the above similarity loss, with d the representation dimensionality. The intuition is that optimizing this loss should push together the embeddings of similar trajectories and push apart the embeddings of dissimilar trajectories. Before training the representation with the loss in (6.2), we may also pre-train it using unsupervised learning [156], which we experiment with in Sec. 6.3.

6.2.3 Using SIRL for Reward Learning

Given a learned embedding ϕ, we can use it for learning models of specific user preferences. While we focus on learning from pairwise preferences, we note that ϕ can in principle be used in downstream tasks that learn from many types of human feedback [148]. When learning a reward function from human preferences, we show the human two trajectories, ξ_A and ξ_B, and then ask which of these two the human prefers. We collect a data set of such preferences $\mathcal{D}_{pref} = \{(\xi_A^i, \xi_B^i, \ell^i)\}$ where $\ell^i = 1$ if ξ_A^i is preferred to ξ_B^i, denoted $\xi_A^i \succ \xi_B^i$, and $\ell^i = 0$ otherwise. We interpret the human's preferences through the lens of the Bradley-Terry preference model [47]:

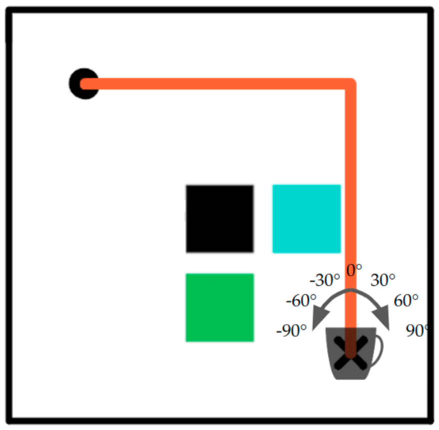
$$P_\theta(\xi_A \succ \xi_B) = \frac{e^{R_\theta(\phi(\xi_A))}}{e^{R_\theta(\phi(\xi_A))} + e^{R_\theta(\phi(\xi_B))}} \ . \tag{6.3}$$

We learn the reward function with a simple cross-entropy loss:

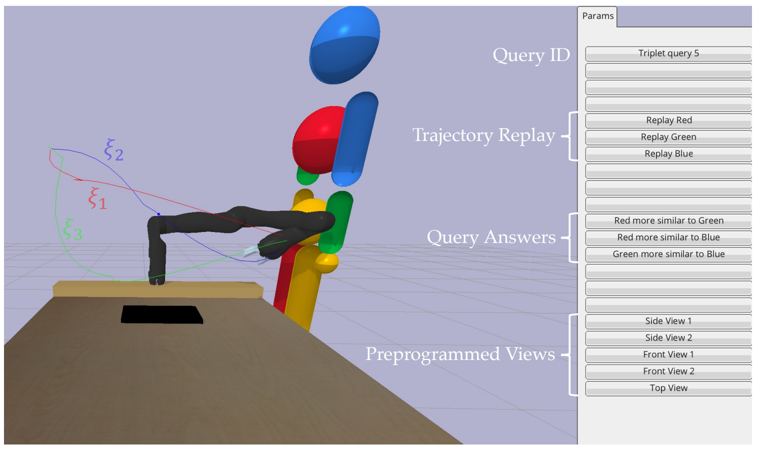
$$\mathcal{L}_{pref}(\theta) = -\sum_{i=0}^{|\mathcal{D}_{pref}|} \ell^i \log P_\theta(\xi_A^i \succ \xi_B^i) + (1 - \ell^i) \log P_\theta(\xi_B^i \succ \xi_A^i) \ . \tag{6.4}$$

6.2.4 Adapting to Different User Preferences

We want robots that can adapt to changes to an individual user's preferences depending on the context as well as quickly adapt to new users' preferences. Rather than learn each preference independently by collecting a new set of human data and training a completely new reward function R_θ, we study whether we can leverage the latent space learned by SIRL to perform more accurate and sample-efficient multi-preference learning. When learning a new user's preference model, R_θ the robot can use ϕ to more quickly learn the reward function $R_\theta(\phi(\xi))$. Our main idea is that because this shared latent representation ϕ is trained via preference-agnostic similarity queries, it is more transferable than using a multi-task or meta-learning approach, where the pre-trained network is trained using multiple, specific task objectives. Furthermore, because SIRL uses human input to train ϕ, we hypothesize that the learned feature space will be better suited for learning human reward functions than a latent space learned via unsupervised training.



(a) GridRobot environment.



(b) JacoRobot environment and user study interface.

Figure 6.2: Visualization of the experimental environments.

6.3 Experiments: Learning Generalizable Representations

We investigate the SIRL representations and their benefits for preference learning using simulated human input in two environments with ground truth rewards and features.

6.3.1 Environments

GridRobot (Fig. 6.2a) is a 5-by-5 gridworld with two obstacles and a laptop (the blue, green, and black boxes). Trajectories are sequences of 9 states with the start and end in opposite corners. The 19-dimensional input consists of the x and y coordinates of each state and a discretized angle in $\{-90°, -60°, -30°, 0°, 30°, 60°, 90°\}$ at the end state. The simulated human answers queries based on 4 features ϕ^* in this world: Euclidean distances to each object, and the absolute value of the angle orientation.

JacoRobot (Fig. 6.2b) is a pybullet [74] simulated environment with a 7-DoF Jaco robot arm on a tabletop, with a human and laptop in the environment. Trajectories are length 21, and each state consists of 97 dimensions: the xyz positions of all robot joints and objects, and their rotation matrices. This results in a 2037-dimensional input space, much larger than for GridRobot. The 4 features of interest ϕ^* for the simulated human are: a) *table* — distance of the robot's End-Effector (EE) to the table; b) *upright* — EE orientation relative to upright, to consider whether objects are carried upright; c) *laptop* — xy-plane distance of the EE to a laptop, to consider whether the EE passes over the laptop at any height; d) *proxemics* [212] — proxemic xy-plane distance of the EE to the human, where the EE is considered closer to the human when moving in front of the human that to their side.

In GridRobot the state space is discretized, so the trajectory space Ξ can be enumerated; however, the JacoRobot state space is continuous, so we construct Ξ by smoothly

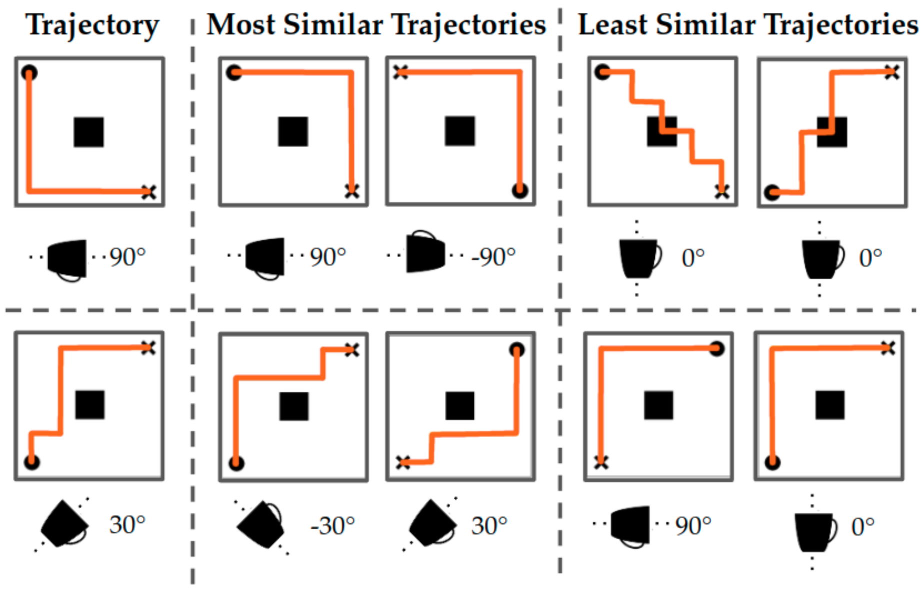


Figure 6.3: SIRL picks the two trajectories that are most and least similar to a query trajectory. Top: trajectories are similar in features despite being dissimilar in states. Bottom: trajectories are dissimilar in features despite being close in states.

perturbing the shortest path trajectories from 10,000 randomly sampled start-goal pairs (see Sec. 6.6.1). We generate similarity and preference queries by randomly sampling from Ξ. The simulated human answers similarity queries by computing the 4 feature values for each of the three trajectories and choosing the two that were closest in the feature space. For preference queries, the simulated human computes the ground truth reward and samples the trajectory with the higher reward. The space of true reward functions (used to simulate preference labels) is defined as linear combinations of the 4 features described above. The robot is not given access to the ground-truth features nor the ground-truth reward function but must learn them from similarity and preference labels over raw trajectory observations.

6.3.2 Qualitative Examples

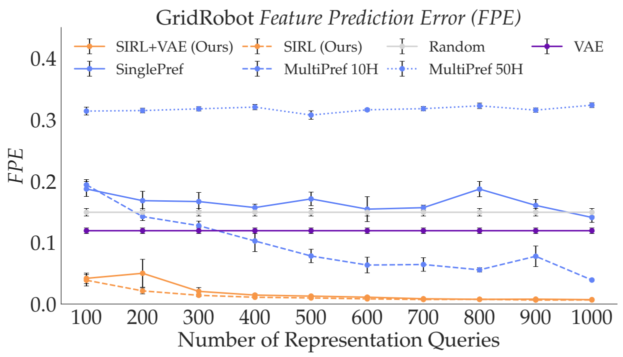
In Fig. 6.3 we show similar and dissimilar trajectories learned by SIRL in a simplified GridRobot environment with only the laptop and the joint angle. *Top*: the given trajectory stays far from the laptop and holds the cup on its side; SIRL learns that trajectories that share those features are similar, despite being dissimilar in the state-space. *Bottom*: the trajectory stays close to the laptop and holds the cup at an angle; SIRL learns that trajectories that hold the cup upright and stay far from the laptop are dissimilar, despite being similar in the state-space (going up and then right).

6.3.3 Experimental Setup

Manipulated Variables. We test the importance of user input that is designed to teach the representation by comparing SIRL with multi-task learning techniques from generic preference queries, and unsupervised representation learning. We have 4 baselines: a) ***VAE***, which learns a representation with a variational reconstruction loss [156]; b) ***MultiPref***, a multi-task baseline [109], where we learn the representation ϕ implicitly by training multiple reward functions (each with shared initial layers) via preference learning; c) ***SinglePref***, a hypothetical method that learns from an ideal user who weighs all features equally; d) ***Random***, a randomly initialized embedding, which does not benefit from human data but is also immune from any spurious correlations that might be learned from biased data. For MultiPref, we trained versions with 10 and 50 simulated human preference rewards for good coverage of the reward space. All embeddings have the same network size: for GridRobot we used MLPs with 2 layers, 128 units each, mapping to 6 output neurons, while for JacoRobot we used 1024 units to handle the larger input space (see Sec. 6.6.2). For a fair comparison, we gave SIRL, SinglePref, and MultiPref equal amounts of human data for pre-training: N similarity queries for SIRL, and N preference queries (used for a single human for SinglePref or equally distributed amongst humans for MultiPref). We also performed ablations with and without VAE pre-training and found that SinglePref and MultiPref are better without VAE (see Sec. 6.6.3).

Dependent Measures. To test the quality of the learned representations, we use two metrics: *Feature Prediction Error (FPE)* and *Test Preference Accuracy (TPA)*. The ***FPE*** metric is inspired by prior work that argues that good representations are linearly separable [73, 169, 243]. Our goal is to measure whether the embeddings contain the necessary information to recover the 4 ground-truth features in each environment. We generate data sets of sampled trajectories labeled with their ground truth (normalized) feature vector $\mathcal{D}_{FPE} = \{\xi, \phi^*\}$. We freeze each embedding and add a linear regression layer on top to predict the feature vector for a given trajectory. We split $\mathcal{D}_{FPE}$ into 80% training and 20% test pairs, and *FPE* is the mean squared error (MSE) on the test set between the predicted feature vector and the ground truth feature vector. For the human query methods, we report *FPE* with increasing number of representation training queries N.

For ***TPA***, we test whether good representations necessarily lead to good learning of general preferences. We use the trained embeddings as the base for 20 randomly selected test preference rewards. For each R_{θ_i}, we generate a set of labeled preference queries $\mathcal{D}^{\theta_i}_{pref} = \{\xi_A, \xi_B, l\}$, which we split into 80% for training and 20% for test. We train each reward model with M preference queries per test reward, and we vary M. All preference networks have the same architecture: we take the embedding ϕ pre-trained with the respective method, and add new fully connected layers to learn a reward function from trajectory preference labels. For GridRobot we used MLPs with 2 layers of 128 units, and for JacoRobot we used 1024 units. We found that all methods apart from SIRL worked better with unfrozen embeddings (Sec. 6.6.3). We report TPA as the preference accuracy



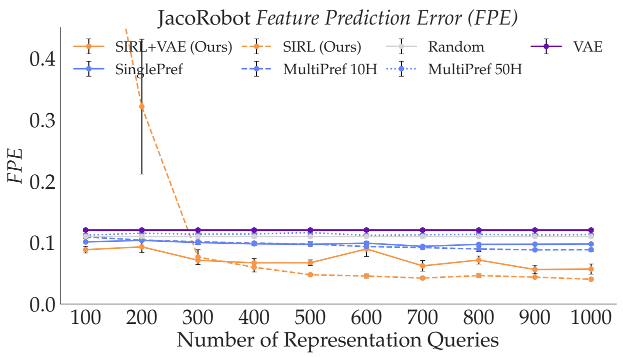


Figure 6.4: *FPE* for the GridRobot (left) and JacoRobot (right) environments with simulated human data. With enough data, SIRL learns representations more predictive of the true features ϕ^* in both simple and complex environments.

for the learned reward models on the test preference set, averaged across the test human preferences.

Hypotheses. We test two hypotheses:
H1. Using similarity queries specifically designed to teach the representation (SIRL) leads to representations more predictive of the true features (lower *FPE*) than unsupervised (VAE), implicit (MultiPref, SinglePref), or random representations.
H2. SIRL representations result in more generalizable reward learning (higher *TPA*) than unsupervised (VAE), implicit (MultiPref, SinglePref), or random representations.

6.3.4 Results

In Fig. 6.4 we show *FPE* for both environments with varying representation queries N from 100 to 1000. For GridRobot, both versions of SIRL (with or without VAE pre-training) perform similarly and outperform all baselines, supporting H1. When pre-training with preference queries, MultiPref with 10 humans performs better than SinglePref or MultiPref with 50 humans: SinglePref may be overfitting to the one human preference it has seen, while when MultiPref has to split its data budget among 50 humans it ends up learning a worse representation than Random. There is a balance to be struck between the diversity in human training rewards covered and the amount of pre-training data each reward gets, a trade-off which SIRL avoids because similarity queries are agnostic to the particular human reward. For the more complex JacoRobot, both versions of SIRL outperform all baselines, in line with H1, although SIRL without VAE scores better than with it.

In Fig. 6.5 we present the *TPA* score for both environments with a varying amount of test preference queries M from 10 to 190, and N = 100, 500, and 1000. For GridRobot, each respective method performs comparably with different Ns, suggesting that this is a simple enough environment that low amounts of representation data are sufficient. For JacoRobot, this is not the case: with just 100 queries, SIRL with VAE pre-training performs

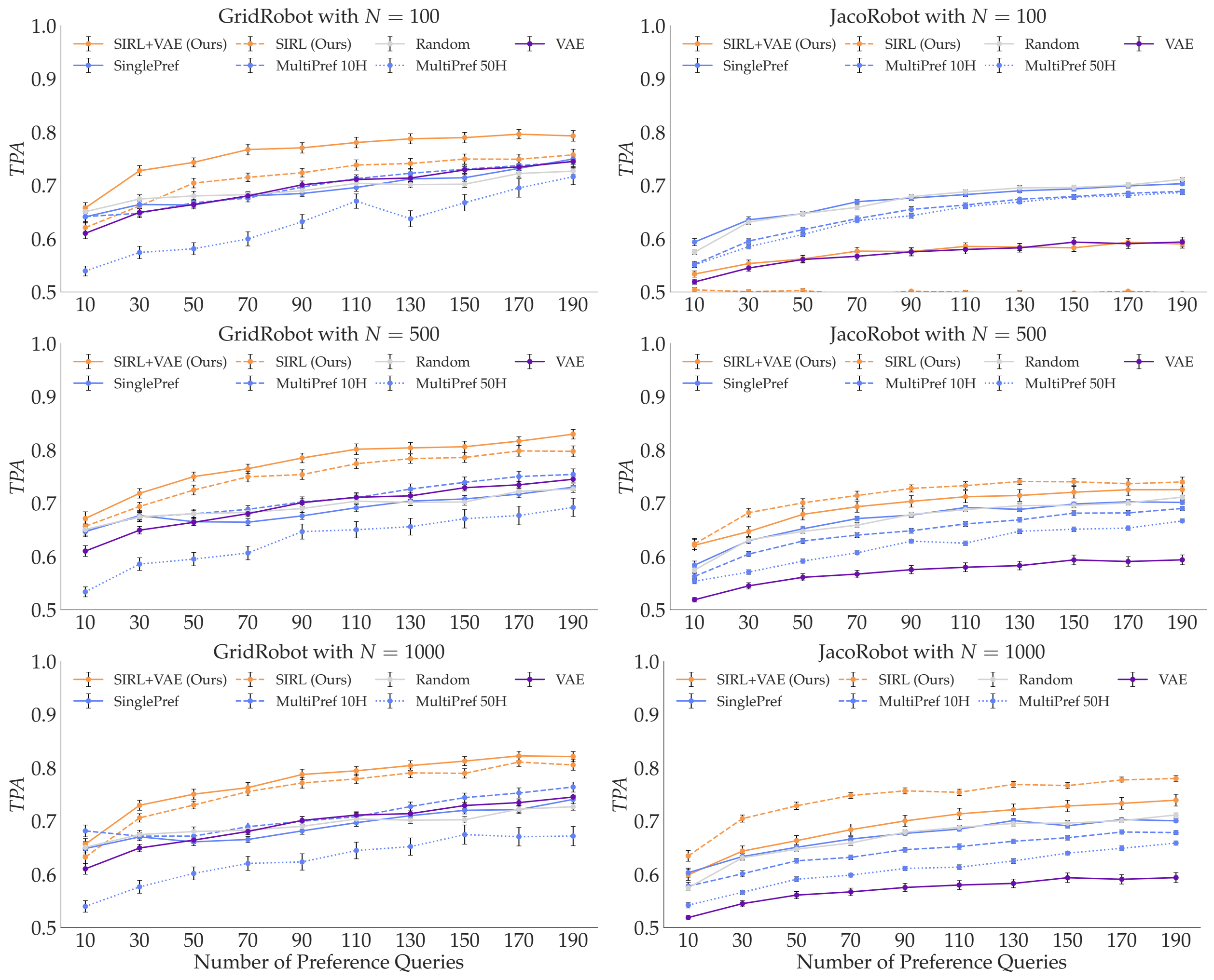


Figure 6.5: *TPA* for GridRobot (left) and JacoRobot (right) with simulated data. With enough data, SIRL recovers more generalizable rewards than unsupervised, preference-trained, or random representations.

like VAE, SIRL without pre-training has random performance (since it's frozen), and the preference baselines all perform close to Random, as if they weren't trained with queries at all. For larger N, both versions of SIRL start performing better than the baselines, suggesting that with enough data a good representation can be learned.

Focusing on $N = 1000$, our results support H2: both SIRLs outperform all baselines in both environments, although for JacoRobot SIRL without VAE is better than with VAE. In the GridRobot environment VAE pre-training helps SIRL. However, while VAE performs comparably to other baselines in GridRobot, it severely underperforms in JacoRobot. This suggests that the reconstruction loss struggles to recover a helpful starting representation when the input space is higher dimensional and correlated. As a result, using the VAE pre-training to warmstart SIRL hinders performance when compared to starting from a blank slate. When comparing the preference-based baselines, in GridRobot they all perform similarly apart from MultiPref with 50 humans. In JacoRobot we see a trend that more preference humans does not necessarily result in better performance. This confirms our observation from Fig. 6.4 that deciding on an appropriate number of human preferences to use for multi-task pretraining is challenging, a problem that SIRL bypasses.

Summary. With enough data, SIRL learns representations more predictive of the true features (H1), leading to learning more generalizable rewards (H2). This does not necessarily mean that SIRL representations are perfectly aligned with causal features — they are just *better* aligned, so the learned rewards are also better. When VAE pre-training recovers sensible starting representations it further reduces the amount of human data SIRL needs, otherwise it hurts performance. Lastly, surprisingly, Random is often better than pre-training with preference queries: preference-based methods may learn features that correlate with the training data but are not necessarily causal, and an incorrectly biased representation is worse for learning downstream rewards than starting from scratch.

6.4 User Study: A Proxy Similarity Task for People

We now present a user study with novice users that provide similarity queries via an interface for the JacoRobot environment.

6.4.1 Experiment Design

We ran a user study in the JacoRobot environment, modified for only two features: *table* and *laptop* (we removed the humanoid in the environment). We designed an interface where people can click and drag to change the view, and press buttons to replay trajectories and record their query answer (Fig. 6.2b). We chose to display the Euclidean path of each trajectory in the query traces, as we found that to help users more easily compare trajectories to one another.

The study has two phases: collecting similarity queries and collecting preference queries. In the first phase, we introduce the user to the interface and describe the two features of interest. Because similarity queries are preference-agnostic, we describe examples of possible preferences akin to the ones in the introduction of the chapter, but we do not bias the participant towards any specific preference yet. Each participant practices answering a set of pre-selected, unrecorded similarity queries, and then answers 100 recorded similarity queries. In the second phase, we describe a scenario in the environment that has a specific preference associated with it (e.g. "There's smoke in the kitchen, so the robot should stay high from the table" or "There is smoke in the kitchen and the robot's mug is empty, so you want to stay far from the table and close to the laptop.") and assign different preference scenarios to each participant. Each person practices unrecorded preference queries, then answers 100 preference queries.

Participants. We recruited 10 users (3 female, 6 male, 1 non-binary, aged 20-28) from the campus community. Most users had technical background, so we caution that our results will speak to SIRL's usability with this population rather than more generally.

Manipulated Variables. Guided by the results in Fig. 6.5, we compare our best performing method, SIRL without VAE, to Random, the best performing realistic baseline. For SIRL we collect 100 similarity queries from each participant and train a shared representation using all of their data.

Dependent Measures. We present the same two metrics from Sec. 6.3, *FPE* and *TPA*. For *TPA*, we collect 100 preference queries for each user's unique preference, we use 70% for training individual reward networks which we evaluate on the remaining 30% queries (Real). We compute *TPA* with cross-validation on 50 splits. To demonstrate how well SIRL works for new people who don't contribute to learning the similarity embedding, we also train SIRL on the similarity queries of 9 of the users and compute *TPA* on the held-out user's preference data (Held-out), for each user, respectively. Lastly, because real data tends to be noisy, we also compute *TPA* with 70 simulated preference queries for 10 different rewards, which we also evaluate on a simulated test set (Simulated).

Hypotheses. Our hypotheses for the study are:
H3. Similarity queries (SIRL) recover more salient features than a random representation (lower *FPE*), even with novice user data.
H4. The SIRL representation results in more generalizable reward learning (higher *TPA*), even with novice similarity queries.

6.4.2 Analysis

Fig. 6.6 summarizes the results. On the left, SIRL recovers a representation twice as predictive of the true features, supporting H3. A 2-sided t-test ($p < .0001$) confirms this. This suggests SIRL can recover aspects of people's feature representation even with noisy similarity queries from novice users. On the right (Real), SIRL recovers more generalizable rewards on average than Random, providing evidence for H4. Furthermore, using the SIRL representation on a novel user (Held-out) also performs better than Random, and the result appears almost indistinguishable from Real. This suggests that similarity queries

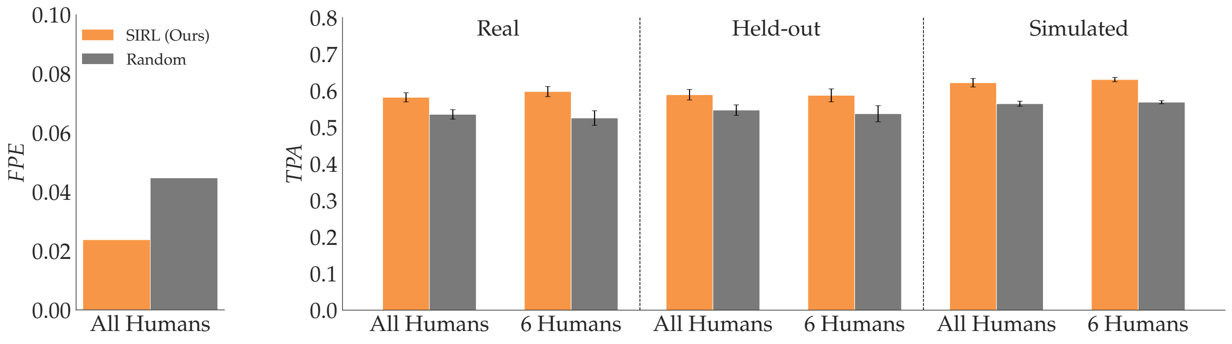


Figure 6.6: Study values for *FPE*, and *TPA* with real and simulated preferences. Even with novice similarity queries, SIRL recovers representations both more predictive of the true features and more useful for learning different user rewards than the baseline.

can be effectively crowd-sourced and the resulting representation works well for novel user preferences. Lastly, training with simulated preference queries slightly improves performance for both methods, suggesting that noise in the human preference data can be substantial. Three ANOVAs with method as a factor find a significant main effect (F(1, 18) = 6.0175, p = .0246, F(1, 18) = 4.7547, p = .0427, and F(1, 18) = 16.1068, p < .001, respectively). For each of the 3 cases, we also separated the 6 humans that were assigned preferences pertaining to both features (e.g. "There is smoke in the kitchen and the robot's mug is empty, so stay far from the table and close to the laptop."). SIRL performance is slightly better when using preference data from this subset of users, hinting that perhaps the learned representation entangled the two features.

Overall, the quantitative results support H3 and H4, providing evidence that SIRL can recover more human-aligned representations. Subjectively, some users found the 2D interface deceiving at times, as they would judge trajectory similarity differently based on the viewpoint. This is a natural artifact of visualizing a 3D world in 2D, but future work should investigate better interfaces. Some users reported struggling to trade off the two features when comparing trajectories. This is in part due to the fact that we almost "engineered" their internal representation, so a more in-the-wild study could determine whether similarity queries are indeed preference-agnostic. Lastly, some queries were easier than others: users' time-to-answer varied across triplets suggesting that future work could use it as a confidence metric for how much to trust their answer.

6.5 Discussion

In this chapter, our goal was to tackle the problem of learning good representations that capture the features that matter, while excluding spurious features. If we had such a representation, then learning rewards that capture different preferences and tasks on top of it would lead to generalizable models that reliably incentivize the right behavior across different situations, rather than picking up on correlates and being unable to distinguish good from bad behaviors on new data. Our idea was to implicitly tap into this representation by asking people what they find similar versus not, because two behaviors will be similar if and only if their representations are similar. We introduced a novel human input type, trajectory similarity queries, and tested that it leads to better representations than those learned through self-supervision or via multi-task learning: it enables learning rewards from the same training data that better rank behaviors on test data.

That said, we need to be explicit that this is not the be-all end-all solution to our goal above. The representations learned, as we see in simulation, are not perfectly aligned with causal features — they are just *better* aligned. The learned rewards are not perfect — they are just better than alternatives. Similarity queries do not solve the problem fully, potentially because they suffer from the same issue preference queries do: when multiple important features change, it becomes harder to make a judgement call on what is more similar. The advantage that we see from similarity queries, though, is that rewards for particular tasks might ignore or down-weigh certain features that matter in other tasks, while similarity queries are task-agnostic and implicitly capture the distribution of tasks in the human's head. Rather than having to specify a task distribution for multi-task learning, with similarity queries we are (implicitly) asking the user to leverage their more general-purpose representation of the world. But thinking about how to overcome the challenge that multiple changing features make these queries harder to answer opens the door to exciting ideas for future work. For instance, what if we iteratively built the space, and based similarity queries on some current estimate of what are the important features; over time, as the representation becomes more aligned with the human's, the queries would get better at honing in on specific features.

Another obvious limitation is that we did not do an in-the-wild study. In theory, similarity queries should be used when people already have a robot they are familiar with and, thus, have a distribution of tasks they care about in their everyday contexts, but in our study we needed to explain to users these contexts and what might be important. In doing so, we almost "engineered" their internal representation. As robots become more prevalent, a follow-up study where users are given much less structure and allowed to actually tap into *their* unaltered representation might be possible.

In a sense, with SIRL we build a foundation model, and this may require hundreds of queries to learn a good representation. While we don't think having this much data when pre-training is unreasonable, especially since it leads to significant desirable performance improvement over baselines, sample-complexity is crucial to address as we scale to more complex robotic tasks. Because similarity queries are task agnostic, we can crowd-source the queries from multiple people (as we did in the study) and rely on this economy at scale to alleviate user burden. Future work could also look at active querying methodologies to ask the person for more informative similarity queries and reduce data amounts.

A further avenue of work is extending beyond reward learning, using SIRL representations directly for learning policies or exploration functions. We emphasize that similarity queries are not a replacement for self-supervised learning; rather, we view them as complementary — self-supervised learning might be able to leverage expert-designed heuristics to eliminate many of the spurious features, while SIRL might serve to fine-tune the representation. How to properly combined the two remains an open question.

Overall, similarity queries are a step towards recovering human-aligned representations. They improve upon the state of the art, and can benefit from further exploration in how to combine them with other inputs and self-supervision, and how to make them easier through better interfaces and query selection algorithms.

6.6 Additional Details and Comparisons

6.6.1 Trajectory Generation

In GridRobot the state space is discretized, so the trajectory space Ξ can be enumerated; however, the JacoRobot state space is continuous, so we need to construct Ξ by sampling the infinite-dimensional trajectory space. We randomly sample 10,000 start-goal pairs and compute the shortest path in the robot's configuration space for each of them, ξ^{SG}. Each trajectory has horizon H and consists of n-dimensional states. We then apply random torque deformations u to each trajectory to obtain a deformed trajectory ξ^{SG}_D. We randomly select up to 3 states along the trajectory, and then deform each of the selected states with a different random torque u. To deform a trajectory in the direction of u we follow:

$$\xi^{SG}_D = \xi^{SG} + \mu A^{-1}\tilde{u} \quad , \tag{6.5}$$

where $A \in \mathbb{R}^{n(H+1)\times n(H+1)}$ defines a norm on the Hilbert space of trajectories and dictates the deformation shape [88], $\mu > 0$ scales the magnitude of the deformation, and $\tilde{u} \in \mathbb{R}^{n(H+1)}$ is u at indices nt through $n(t+1)$ and 0 otherwise ($\tilde{u}$ is 0 outside of the chosen deformation state index). For each deformation, we randomly generated μ and the index of the state the deformation is applied to. For smooth deformations, we used a norm A based on acceleration, but other norm choices are possible as well (see Dragan et al. [88] for more details). We took inspiration for this deformation strategy from Bajcsy et al. [25].

6.6.2 Training Details

We describe architecture and optimization details for reproducibility.

6.6.2.1 Feature Networks. All embeddings have the same network size: for GridRobot we used MLPs with 2 hidden layers, 128 units each, mapping to 6 output neurons, while for JacoRobot we used 1024 units to handle the larger input space. For both environments, we used ReLU non-linearities after every linear layer.

We trained the VAE network with a standard variational reconstruction loss [156] also including a KL-divergence-based regularization term (to make the latent space regular).

The regularization part of the loss had a weight of $\lambda = 0.01$. For both environments, we optimized the loss function using Adam for 2000 epochs with an exponentially decaying learning rate of 0.01 (decay rate 0.99999) and a batch-size of 32.

SinglePref and MultiPref with 10 and 50 humans are trained using the standard preference loss in (6.4). Christiano et al. [71] ensured that the rewards predicted by the preference network remain small by normalizing them on the fly. We instead add an l_2 regulatization term on the predicted reward to the preference loss, with a weight of 10 for GridRobot and 1 for JacoRobot. All three methods optimize this final loss in the same way: for GridRobot, we use Adam for 5000 epochs with a learning rate of 0.01 and batch-size 32, while for JacoRobot a lower learning rate of 0.001 results in more stable training.

Lastly, for SIRL we had the option to first pre-train with the VAE loss. Training with the similarity objective in (6.2) happens disjointly, after pre-training. For both GridRobot and JacoRobot, we optimized this loss function using Adam for 3000 epochs with an exponentially decaying learning rate of 0.004 (decay rate 0.99999) and batch-size 64.

We note that our current architectures assume fixed-length trajectories but one could adopt an LSTM-based architecture for trajectories of varying length [267].

6.6.2.2 Preference Networks. For evaluating *TPA*, we used preference networks on top of each tested embedding. For GridRobot we used MLPs with 2 hidden layers of 128 units, and for JacoRobot we used 1024 units for larger capacity. For both environments, we used ReLU non-linearities after every linear layer. We added the same l_2 regularization to the loss in (6.4) as before, with weight 10 for GridRobot and 1 for JacoRobot. For GridRobot, we optimized our final loss function using Adam for 500 epochs with a learning rate of 0.001 and a batch-size of 64. For JacoRobot, we increased the number of epochs to 1000.

6.6.3 Ablations

Fig. 6.5 illustrates results with frozen SIRL, and unfrozen baselines without VAE pre-training, as these were the best configurations we found for each method. In this section, we show the complete ablation we performed to decide which methods benefit from frozen or unfrozen embeddings, or VAE pre-training. Fig. 6.7 showcases the result of this ablation on both GridRobot and JacoRobot. Overall, we see that SIRL does better when the learned representation is frozen, while all the other methods do better when the representations is unfrozen. SinglePref and the MultiPref baselines perform better without VAE pre-training, while SIRL sometimes benefits from pre-training in simple environments like GridRobot.

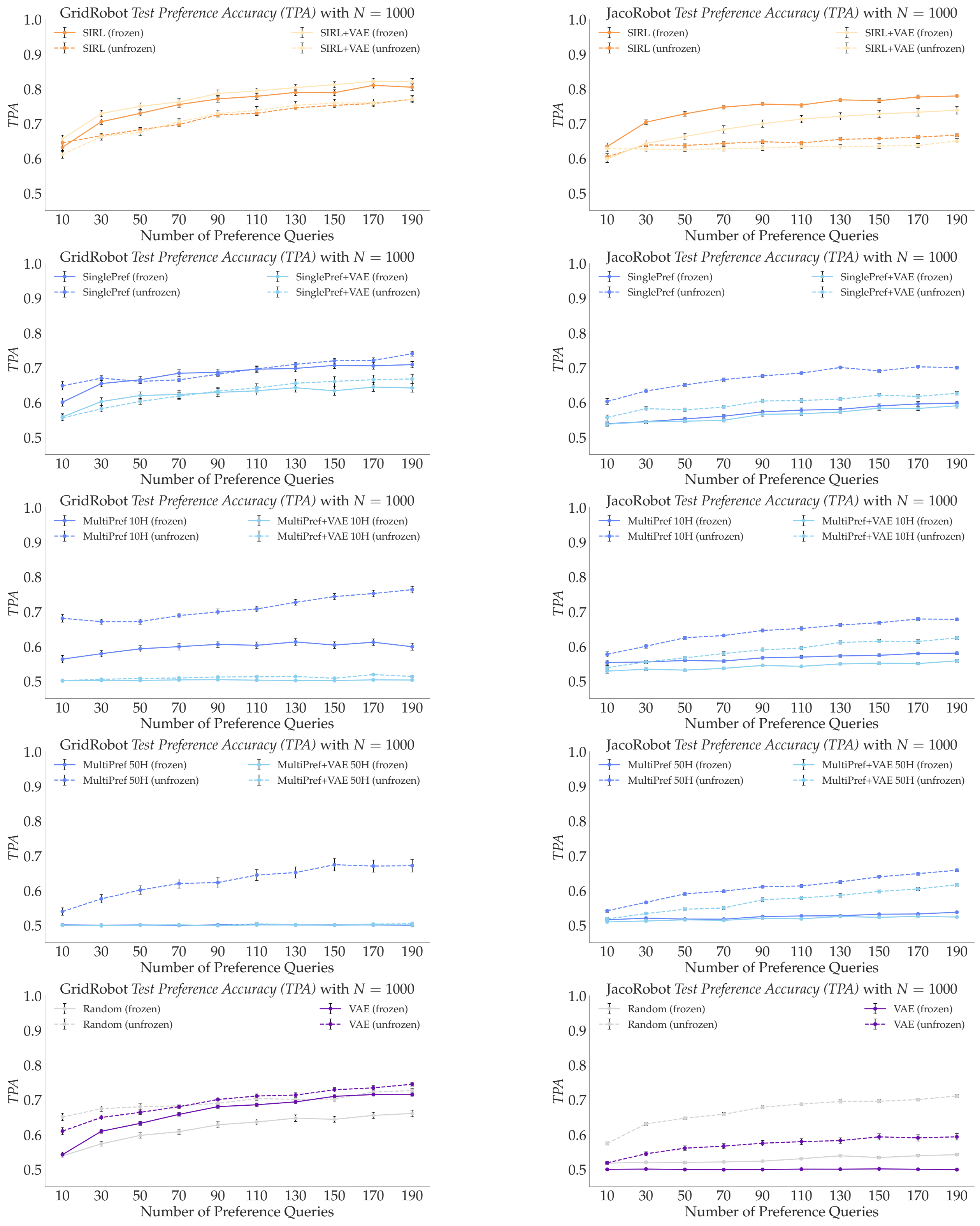


Figure 6.7: Ablation results for GridRobot (left) and JacoRobot (right). Overall, SIRL does better when the learned representation is frozen, while all other method do better when the representations is unfrozen. SinglePref and the MultiPref baselines perform better without VAE pre-training, while SIRL sometimes benefits from pre-training in simple environments like GridRobot.

Part III

Detecting Representation Misalignment

Human behavior is not random – it's intentional and reflective of internal goals and preferences. To interpret and learn from it, robots rely on a *model* of how people act: they often assume humans trade off what matters for the task – the representation – and behave *rationally* to achieve that trade-off. If the robot expects representations to be naturally aligned, it interprets the human's actions as rationally trading off *its* representation instead of their own. Thus, with the wrong representation, the robot may incorrectly anticipate what the person will do or misinterpret their guidance for how to do the task, resulting in undesired behaviors and poor coordination. In Part III we enable the robot to instead automatically detect misalignment during real-time interaction with the human. The idea here is to jointly estimate the human's intent *and* their rationality in achieving it, and reinterpret that rationality as a *confidence* in how much the robot should trust its representation: *if the person's behavior seems irrational under the robot's current representation, then that representation must be misaligned*. This provides the robot with a fast and principled way to monitor misalignment, and it can even overcome it: once the robot knows its representation is wrong, it can re-learn or expand it to be aligned with the human's (like in Part II). We demonstrate that the robot can still learn from behavior it understands, but is now robust to misinterpretation and ultimately learns what humans want. Lastly, we show our insight applies broadly to many types of human data like physical demonstrations or corrections (Chapter 7) and teleoperation in shared autonomy (Chapter 8).

Chapter 7

Misalignment Detection in Physical HRI

This chapter is based on the papers "Quantifying Hypothesis Space Misspecification in Learning from Human–Robot Demonstrations and Physical Corrections" [38] and "Learning under Misspecified Objective Spaces" [43], written in collaboration with Andrea Bajcsy, Jaime Fisac, Sampada Deglurkar, and Anca Dragan.

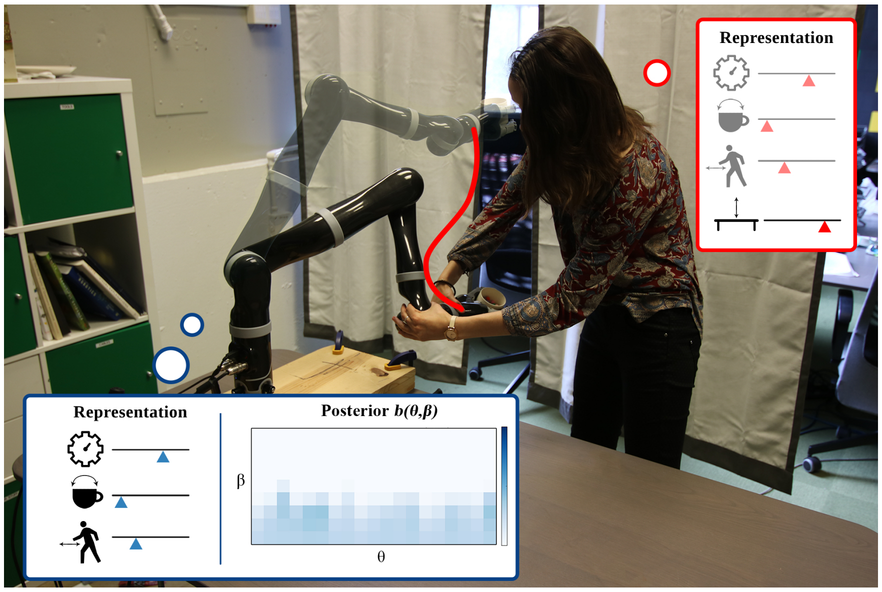


Figure 7.1: A household robotics scenario where the person physically interacts with the robot. The person prefers the robot to keep cups closer to the table, but accounting for the table (outside of collisions) is not in the robot's representation for what the person might care about. Thus, the robot's internal situational confidence, β, about what the human input means is low for all hypothesised reward parameters θ.

Autonomous systems are increasingly interfacing and collaborating with humans in a variety of contexts, such as semi-autonomous driving, automated control schemes on airplanes, or household robots working in close proximity with people. While the improving capabilities of robotic systems are opening the door to new application domains, the substantially greater complexity and interactivity of these settings makes it challenging for system designers to account for all relevant operating conditions and requirements ahead of time. For example, a household robot designer may not know how an end-user would like the robot to interact with the personal possessions in the user's home.

In such situations, it can be beneficial for the robot to use human input as guidance on the desired behavior. In fact, this has enabled researchers and engineers to program advanced behaviors that would have otherwise been extremely challenging to specify. Helicopter acrobatics [1], aggressive automated car maneuvers [161], and indoor navigation [165] are three cases exemplifying the benefit of human input for guiding robot behavior.

To use human input, system designers equip robots with a reward model based on a representation of possible features that the human *could* care about. These can range from quadratic models[162] to complex temporal logic specifications [101] to neural networks [96]. However, anticipating all motivations for human input and specifying a complete representation is challenging. Consider Fig. 7.1 where a human is attempting to change the robot's behavior in order to make it consistently stay close to the table, but the robot's representation of what the human might care about does not include distances to the table. By choosing a class of functions, the system designer implicitly assumes that what the human wants (and is giving input about) can be represented via a member of that class. Unfortunately, when this assumption breaks, the system can misinterpret human guidance, perform unexpected or undesired behavior, and degrade in overall performance.

Two approaches to mitigating this problem could be to either start with a more complex reward model or to continuously increase its complexity given more task data. Unfortunately, even complex models are not guaranteed to encompass all possibilities and re-computing the best model based on human data faces the threat of overfitting to the most recent observations. In contrast, we argue the robot should be able to *understand when it cannot understand the input*. For example, if the end-user in the home is trying to guide the robot to handle fragile objects with care but the system's representation for the space of rewards does not posses a feature for fragility, the robot should deduce that this input cannot be well explained by any of its given reward hypotheses.

In this chapter, we formalize how autonomous systems can explicitly reason about how well they can explain given human inputs. To do this, we observe that if a human input appears unlikely with respect to all possible reward hypotheses, then the robot's reward space, or representation, is misaligned. We build on previous work centered around this observation to propose a Bayesian inference framework focused on inferring both reward parameters and their corresponding *situational confidence*. If the robot is in situations like Fig. 7.1 where none of the reward hypotheses explain the human's input well, then the situational confidence will be low for all hypotheses, indicating that the robot's representation is not sufficiently rich to understand the human's input. However, when the robot's representation is well specified, our framework does not impede the robot from inferring the correct task objectives — in fact, the situational confidence will be high, providing an indicator of how well the system can understand the objective.

We illustrate the utility of situational confidence estimation in quantifying representation misalignment for two types of human input: demonstrations and corrections. In Chapter 8 we will demonstrate that the principles outlined in our formalism are more general than just these two domains and have implications for a wider range of interaction modes, including shared autonomy. Our contributions in this section are:

1. we introduce a general framework for quantifying representation misalignment when the human and the robot are acting on the same dynamical system;

2. we showcase the framework for learning from demonstrations using user demonstration data for an arm motion planning task;

3. we showcase the framework for learning from physical corrections by deriving an algorithm for online (close to real-time) inference and testing it in a user study.

This chapter is organized as follows: Sec. 7.1 places this work in the context of existing literature on robots learning from humans and confidence estimation. Sec. 7.2 frames the confidence estimation problem more formally for scenarios where the human and robot operate on the same dynamical system. Sec. 7.3 directly instantiates the framework in Sec. 7.2 for the case of learning from demonstrations. Sec. 7.4 presents a derivation of approximations of the general formalism for tractable online inference from human corrections. Sec. 7.5 showcases our proposed approach in several case studies where the robot's representation cannot or only partially explain the human's input. Sec. 7.6 presents the results of a user study of our approach as applied to a 7-DoF robotic manipulator learning from human participants. Sec. 7.7 concludes with a discussion of some of the limitations of our work, as well as suggestions for future research directions.

Overall, we think that the ability to detect misalignment when learning rewards from human input will become increasingly important as robotics capability advances and we will want end-users to customize how the robot behaves. Our work takes a step in this direction by enabling robots to detect when none of the hypotheses they have explain the user input, and our experiments show promising results. Of course, there are still limitations to this. One limitation is in the experiments themselves, which are only for motion planning tasks with low-dimensional representations. A more fundamental limitation is that there will still be cases when the person wants something outside the robot's representation, but the robot can nonetheless explain their current input relatively well with what it has access to, thus confusing misalignment for slight noise in the human input. This will especially be the case as the representation is more expressive, and can only be solved by the robot receiving a lot more human input: each might be explainable by some hypothesis, but eventually no hypothesis can explain all input. More work is needed in studying how to query for diverse human input, as well as how to convey what the robot has learned back to the person, and in general how to have a true collaborative interaction to detect and resolve misalignment in the objective space.

7.1 Prior Work

We group prior work into three main categories: enabling robots to learn from human input, doing so while leveraging uncertainty, and estimating confidence.

7.1.1 Robots Learning from Humans

Programming robots through direct human interaction is a well-established paradigm. Human input can be given to the robot in a variety of forms, from teleoperation of the robot by a user to kinesthetic teaching [17]. In such interaction paradigms, the robot aims to infer a *reward function* or *policy* that best describes the examples that it has received. New avenues of research focus on learning such robot objectives from human input through demonstrations [2, 219], teleoperation data [145], corrections [144, 25], comparisons [71], examples of what constitutes a goal [103], or even specified proxy objectives [123]. In Part III we focus on learning from three such types of human input – demonstrations, physical corrections, and teleoperation – showcasing that the principles outlined in our formalism are generally applicable.

One approach to learning behaviors from human inputs is inverse reinforcement learning (IRL). In classical IRL, the robot receives complete optimal *demonstrations* of how to perform a task, and the robot learns the human's reward function from these observations [151, 214, 219]. In this paradigm, it is typically assumed that the expert is trying to optimize an unknown reward function. The robot uses the observations of the human's behavior to recover the underlying objective.

Another form of human input are *corrections*: here, the robot performs the task according to how it was programmed and the user corrects aspects of the task to better match their preferences. From these sparse interactions, the robot infers the reward function to improve performance during the next task iteration [262, 236, 152]. Learning from corrections has been explored in offline [144, 118] and online settings [25, 60, 18].

Although powerful, the aforementioned IRL works assume that the human expert provides optimal demonstrations, which is often an unrealistic assumption. Real human input, especially during interaction with high degree-of-freedom systems like robotic manipulators, is noisy and sub-optimal. Second, much of the corrections literature has focused on estimates of the human's objectives. However, in practice, even the most likely estimate might not be a very likely one. Thus, in both domains, we stress that it is important to maintain the uncertainty over the estimated objectives.

7.1.2 Uncertainty in Robot Learning

Rather than estimating a single objective, some learning methods maintain an entire probability distribution over the objective [51, 123, 190, 233]. This not only enables the robot to leverage a prior, but also to then generate its behavior in a way that is mindful of the entire distribution, rather than of just the maximum likelihood estimator.

Bayesian IRL [233] treats demonstrations as evidence about the objective, and does a Bayesian belief update on a prior distribution. Inverse Reward Desing [123] treats the objective a designer specified for a particular set of environments (a "proxy" objective) as evidence about the true desired objective, again obtaining a full distribution over what the designer might want. The intuition is that this observed proxy objective incentivizes behavior that is approximately optimal with respect to the true objective.

Lastly, specifically for input as physical corrections, [190] reasons over the uncertainty of the estimated human preferences through the means of a Kalman filter. The method maintains a mean estimate and a covariance of this estimate as a measure of confidence. These are used in planning the robot's trajectory such that it optimizes for features it is confident about, while avoiding features it is uncertain about.

Although they maintain a full distribution, these works still assume that what the human wants is in the robot's representation. We argue that this is not necessarily a realistic assumption, and later showcase some consequences that arise when it is not true. When the robot's representation is misaligned, even when maintaining uncertainty over the objective, state-of-the-art methods interpret human input as evidence about which hypothesis is correct, rather than considering whether any hypothesis is correct.

7.1.3 Situational Confidence Estimation

Some recent works are studying how to enable robots to understand that their models cannot explain human input well [309, 98, 100]. The authors in [98, 100] employ a noisily-optimal model of human pedestrian motion when the human and the robot operate on separate dynamical systems (and have separate objective functions). The paper introduces the notion of model confidence estimation and uses the apparent likelihood of the human's choice of actions to adjust the confidence in predictions about their behavior.

Our work draws inspiration from the notion of model confidence estimation, generalizing it to the setting of inferring what the robot's objective ought to be. Instead of focusing on misalignment of a discrete set of physical goal locations for pedestrian navigation, here we study misalignment of a relatively complex set of possible robot features in motion planning tasks. As a result of focusing on robot objectives, we also study a different form of human input – that is, input in the context of operating on the same dynamical system, such as full task demonstrations and physical corrections.

7.2 Problem Formulation

We consider a robot R operating in the presence of a human H whom it seeks to assist in the execution of some task. In the most general setting, the robot and the human are both able to affect the state $x \in \mathbb{R}^n$ over time through their respective control inputs:

$$x^{t+1} = f\left(x^t, u_R^t, u_H^t\right) , \tag{7.1}$$

with $u_R \in \mathcal{U}_R$ and $u_H \in \mathcal{U}_H$, where $\mathcal{U}_i$ ($i \in \{H, R\}$) are compact sets. We assume that the human has some consistent preference ordering between different state trajectories and input signals, which could in principle be expressed through a reward function

$$R^*(\mathbf{x}, \mathbf{u}_R, \mathbf{u}_H) \tag{7.2}$$

where the state trajectory is $\mathbf{x} = [x^0, x^1, \dots, x^T] \in \mathbb{R}^{n(T+1)}$, the robot's control input is $\mathbf{u}_R = [u_R^0, u_R^1, \dots, u_R^T] \in \mathbb{R}^{n(T+1)}$, and the human's is $\mathbf{u}_H = [u_H^0, u_H^1, \dots, u_H^T] \in \mathbb{R}^{n(T+1)}$.[1] Note that this hypothesized reward function R^* can be quite general, encoding an arbitrary preference ordering. However, the robot does not in general have access to the human's preferences R^*, and must instead attempt to infer and represent them tractably.

The robot can typically reason over a parametrized approximation of the reward function, which introduces an inductive bias, making inference tractable at the reward of limiting expressiveness: in some cases, the chosen set of parametric functions may fail to encode preferences that would explain the human's behavior with sufficient accuracy. In this work, we will denote by R_θ the reward function induced by parameters $\theta \in \Theta$, and the robot seeks to estimate the human's preferred θ from her control inputs $\mathbf{u}_H$.

In a general setting, since the state trajectory $\mathbf{x}$ is determined not only by the human's actions $\mathbf{u}_H$ but also the robot's $\mathbf{u}_R$, the human would need to reason about how the robot will respond to her decisions. This requires analyzing the interaction in a game-theoretic framework [122, 99], which will not be the object of this work. Instead, we focus on common interaction scenarios in which the robot can approximately assume that the human does not explicitly account for the coupled mutual influence between both agents' decisions. This happens frequently if the human is either providing a demonstration for the robot or intervening to correct the robot's default behavior. In these settings, the typical assumption is that the human has all necessary information about the robot's control input $\mathbf{u}_R$ before deciding on her own $\mathbf{u}_H$.

Thus, given observations of the human input $\mathbf{u}_H$ from an initial state x^0, the robot needs to draw inferences on the reward parameter θ:

$$P(\theta \mid x^0, \mathbf{u}_R, \mathbf{u}_H) = \frac{P(\mathbf{u}_H \mid x^0, \mathbf{u}_R; \theta) P(\theta)}{\int_{\bar{\theta}} P(\mathbf{u}_H \mid x^0, \mathbf{u}_R; \bar{\theta}) P(\bar{\theta}) d\bar{\theta}} \,, \tag{7.3}$$

where $P(\mathbf{u}_H \mid x^0, \mathbf{u}_R; \theta)$ characterizes how the robot expects the human's input to be informed by her preferences, conditioned on the initial state and the robot's controls.

For example, if the human were assumed to act optimally, this model would place all probability on the set of optimal states and actions with respect to the reward R_θ. Of course, this would be an unreasonably strong assumption given that the robot's parametrized reward constitutes a best effort to approximate the human's preferences.

[1] For deterministic dynamics (7.1), having $x^0, \mathbf{u}_R$ and $\mathbf{u}_H$ is enough to fully specify the entire state trajectory $\mathbf{x}$. In this case, the reward function could be rewritten as $R^*(x^0, \mathbf{u}_R, \mathbf{u}_H)$ by implicitly encoding (7.1). For clarity, we use the more general form in (7.2) and make the dependence explicit where needed.

Instead, a useful modeling choice can be to characterize the human as being more *likely* to take actions that are well-aligned with her preferences.

One such model is inspired by the Boltzmann energy-based model satisfying the maximum entropy principle [146]. Following its adaptations as a model of human decision-making in [282, 27, 25], we model the human as a noisily-optimal agent that tends to choose control inputs that approximately maximize the modeled reward:

$$P(\mathbf{u}_H \mid x^0, \mathbf{u}_R; \theta, \beta) = \frac{e^{\beta R_\theta\left(\mathbf{x}(\cdot; x^0, \mathbf{u}_R, \mathbf{u}_H), \mathbf{u}_R, \mathbf{u}_H\right)}}{\int_{\bar{\mathbf{u}}_H} e^{\beta R_\theta\left(\mathbf{x}(\cdot; x^0, \mathbf{u}_R, \bar{\mathbf{u}}_H), \mathbf{u}_R, \bar{\mathbf{u}}_H\right)} d\bar{\mathbf{u}}_H}. \tag{7.4}$$

The inverse temperature coefficient $\beta \in [0, \infty)$ determines the degree to which the robot expects to observe human actions that are consistent with the reward model.

The goal is to detect when the robot does not have a rich enough reward hypothesis space, i.e. when R^* lies far outside of any R_θ. Rather than only interpreting human input as evidence about *which* hypothesis is correct, we additionally focus on considering whether *any* hypothesis is correct. It is thus crucial that the robot can quantify the extent to which any parameter value $\theta \in \Theta$ can correctly explain the observed human input.

7.2.1 Situational Confidence Estimation

The key to our approach goes back to the inverse temperature parameter β in (7.4). Typically, β is a fixed term, encoding the degree to which the robot expects to observe human actions that are optimal. Setting it to 0 models a randomly-acting human, while setting it to ∞ models a perfectly optimal human. However, the possibility of misalignment brings fixing β into question: when the space is correctly specified, we would expect the human actions to indeed be somewhat close to optimal; but when the space is misaligned, *we should expect the actions to be far from optimal for any θ.* Thus, rather than treating β as a fixed term, we build on the work in [98, 100] and explicitly reason over β as an additional inference parameter along with θ. Since β directly impacts the entropy of the human's decision model, it can be used as an effective and computationally efficient measure of the robot's confidence in its parametric interpretation of the human's preference: we say that the robot is assessing its *situational confidence* for the inference task at hand.

Thus, the robot maintains a joint Bayesian belief $b(\theta, \beta)$. For each new measurement of $\mathbf{u}_H$ given $x^0, \mathbf{u}_R$, this belief is updated as:

$$b'(\theta, \beta) = \frac{P(\mathbf{u}_H \mid x^0, \mathbf{u}_R; \theta, \beta) b(\theta, \beta)}{\int_{\bar{\theta}, \bar{\beta}} P(\mathbf{u}_H \mid x^0, \mathbf{u}_R; \bar{\theta}, \bar{\beta}) b(\bar{\theta}, \bar{\beta}) d\bar{\theta} d\bar{\beta}} \ , \tag{7.5}$$

where $b'(\theta, \beta) = P(\theta, \beta \mid x^0, \mathbf{u}_R, \mathbf{u}_H)$.

This inference can be seen as analogous to performing Bayesian Inverse Reinforcement Learning [233] with the Maximum Entropy IRL [311] observation model, where we maintain the full belief instead of just the maximum likelihood estimate, and we explicitly reason over the additional scaling parameter β. By actively performing inference over β, the robot can gain insight into the reliability of its human model in light of new evidence.

Context-dependent Usage of Situational Confidence. How this insight should be used is dependent on the context of the robot's operation. Here, we provide some examples of how situational confidence can be integrated into various human-robot interaction scenarios and robot motion planners.

In collaborative settings where the human and robot are accomplishing a task together (e.g. manipulating an object together), it the robot may stop and ask for clarification from the human whenever sufficient probability mass indicates low confidence:

$$\forall\theta \in \Theta, \arg\max_{\beta} b'(\beta \mid \theta) < \epsilon \quad . \tag{7.6}$$

That is, for a predefined threshold ϵ, if all hypotheses have the most mass on βs lower than ϵ, the robot can raise a flag.

In assistive applications, where the robot is carrying out a task in close physical proximity to the human, the robot may receive intermittent human input to correct it's task performance. In such scenarios, it may be appropriate for the robot to simply dismiss human corrections that it cannot explain in terms of modeled preference parameters and carry on with its pre-defined task. That is, when a human input results in a $b'(\theta, \beta)$ that satisfies (7.6), the input gets discarded.

Situational confidence could also be leveraged by robot motion planners that excel at decision making under uncertainty. Here, the robot may use its joint posterior belief $b'(\theta, \beta)$ to make goal-driven decisions in the presence of the human. The coupling between the inference problem and the robot's planning problem can be viewed as a partially observable Markov decision process (POMDP), where the hidden parts of the state are the reward parameter θ and the situational confidence β, the robot receives observations about them via human actions $\mathbf{u}_H$, it takes actions $\mathbf{u}_R$, and it optimizes an unknown parametrized reward R_θ. Our problem is, thus, akin to identifying misalignment in the state space of the POMDP. However, inference and planning in such spaces requires solving the full POMDP, which is computationally intractable for large, real-world problems [149].

Alternative, less computationally demanding motion planning approaches are also amenable to our framework, where the robot plans to maximize the expected reward for the human given its current belief, by marginalizing over β:

$$\max_{\mathbf{u}_R} \mathop{\mathbb{E}}_{\theta\sim b} \left[R_\theta(\mathbf{x}, \mathbf{u}_R, \mathbf{u}_H)\right] \quad , \tag{7.7}$$

for an expected human input $\mathbf{u}_H$ that will typically be $\mathbf{0}$ if the robot is attempting to perform the task without the need for active human intervention. To understand the implication of (7.7) as a function of β, we need to understand the posterior belief marginalized over β that we are taking the expectation over. At one extreme, if for all θs the conditional distribution $b'(\beta \mid \theta)$ puts all probability mass on $\beta = 0$ (i.e. input poorly explained), since

$P(\mathbf{u}_H \mid x^0, \mathbf{u}_R; \theta, \beta = 0)$ is the same for all θs, the robot will obtain a posterior for θ that is equal to the prior. The optimization above becomes the same as optimizing using the robot's prior, i.e. the robot ignores the human input. At the other extreme, if there is one θ that perfectly explains the input and all others do not, the posterior will put all probability mass on that θ, and the robot will switch to optimizing it.

The objective expectation may also be appropriately weighted by the robot's situational confidence for each θ:

$$\max_{\mathbf{u}_R} \mathop{\mathbb{E}}_{\theta,\beta\sim b} \left[\beta R_\theta(\mathbf{x}, \mathbf{u}_R, \mathbf{u}_H)\right] \ , \tag{7.8}$$

which leads the robot to prioritize those components about which it is most certain.

In Sec. 7.3 and Sec. 7.4 we discuss some of these possibilities in the context of learning from demonstrations and corrections.

7.2.2 Reward Representation through Basis Functions

One way to approximate the infinite-dimensional space of possible reward functions using a finite number of parameters is the use of a finite family of basis functions Φ_i[214]. This family can be seen as a truncation of an infinite collection of basis functions spanning the full function space. Parametric approximations R_θ of R^* then have the form

$$R_\theta(\mathbf{x}, \mathbf{u}_R, \mathbf{u}_H) = \sum_{i=1}^{d} \theta^i \Phi_i(\mathbf{x}, \mathbf{u}_R, \mathbf{u}_H) = \theta^T \Phi(\mathbf{x}, \mathbf{u}_R, \mathbf{u}_H) \ . \tag{7.9}$$

Consistent with classical utility theories [282], we further assume that the human's preferences can be approximated through a cumulative return over time, rewriting (7.9) as

$$R_\theta(\mathbf{x}, \mathbf{u}_R, \mathbf{u}_H) = \sum_{i=1}^{d} \theta^i \sum_{t=0}^{T} \phi_i(x^t, u_R^t, u_H^t) \ , \tag{7.10}$$

where $\phi_i : \mathbb{R}^n \times \mathcal{U} \times \mathcal{U} \to \mathbb{R}$ are fixed, pre-specified, bounded real-valued basis functions, θ is the unknown parameter that the robot is trying to fit according to the human's preferences, and d is the dimensionality of its domain Θ.

In the domains presented in Sec. 7.3 and Sec. 7.4, the functions ϕ_i output feature values that encode key aspects of a task—for example distance between the robot body and obstacles in the environment, speed of the motion, or characteristics of a motion planning task. In general, the ϕ_i can either be hand-engineered by a system designer or more generally learned through data-driven approaches [96].

It is important to stress that the misalignment issue we are trying to mitigate is quite general and does not exclusively affect rewards based on hand-crafted features: any representation could ultimately fail to capture the underlying motivation of some human actions. While it may certainly be possible, and desirable, to continually increase the

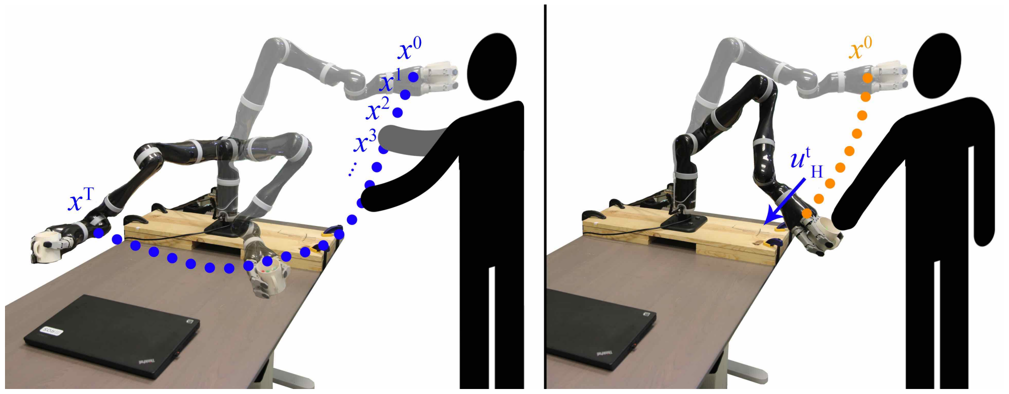


Figure 7.2: (Left) Visual example of a full human-provided demonstration $\mathbf{x}$. (Right) Visual example of a human physical correction u_H^t onto the robot's current trajectory $\mathbf{x}$.

complexity of the robot's representation to capture a richer space of rewards, there will still be a need to account for the presence of yet-unlearned components of the true reward. In this sense, our work is complementary to open-world reward modeling efforts.

Note that using a reward model in the form of (7.10), the observation model (7.4) becomes overparametrized, since for any (θ, β) pair with $\theta \in \Theta$ and $\beta \in [0, \infty)$, one can always find a different $\theta' = c\theta$ with an associated $\beta' = \beta/c$ leading to the same probability distribution over human choices. This is equivalent to using an unrestricted Θ and $\beta = \|\theta\|$. Due to this overparametrization, the absolute value of β does not have a universal meaning, and restricting θ to have a fixed norm is necessary in order to make comparisons between the β values associated to different θ hypotheses. We thus restrict our Θ to the set of vectors with unit norm.

Consider the case where the human provides input for a reward function in the robot's reward space. This results in the robot inferring high probability on the corresponding θ vector on the unit sphere with a high magnitude β. However, if the reward that the human cares about and provides input for is outside the robot's reward space, the robot will infer low probability on all θ vectors in the unit sphere, with low magnitude βs.

7.3 Approach: Misalignment for Demonstrations

7.3.1 Formulation

In learning from demonstrations, the human directly controls the state trajectory $\mathbf{x}$ through her input $\mathbf{u}_H$, which enables her to offer the robot a demonstration of how to perform the task. Fig. 7.2 (left) is an example of such a demonstration.

During the demonstration, the robot is put in gravity compensation mode or is teleoperated, to grant the person full control over the desired trajectory. As such, in this setting, the reward function R_θ does not depend on the robot controls $\mathbf{u}_R$. Additionally, since the person is primarily concerned with the robot's states and not with the (robot or human) actions required to reach those states, we model the human's internal preferences as only dependant on the state trajectory $\mathbf{x}$. Accordingly, the reward function in (7.10) becomes:

$$R_\theta(\mathbf{x}) = \theta^T \Phi(\mathbf{x}). \tag{7.11}$$

The reward does not have a direct dependence on the actions, but it has an indirect one, as $\mathbf{x}$ depends on $\mathbf{u}_R$ and $\mathbf{u}_H$.

In our problem formulation, we would like the robot to explicitly reason about how well it can explain the demonstration given its human model. Thus, we can adapt the model in (7.4) to use this new reward function[2],

$$P(\mathbf{x} \mid \theta, \beta) = \frac{e^{\beta\theta^T\Phi(\mathbf{x})}}{\int_{\bar{\mathbf{x}}} e^{\beta\theta^T\Phi(\bar{\mathbf{x}})} d\bar{\mathbf{x}}} \ , \tag{7.12}$$

then perform the Bayesian update in (7.5)

$$b'(\theta, \beta) = \frac{P(\mathbf{x} \mid \theta, \beta) b(\theta, \beta)}{\int_{\bar{\theta}, \bar{\beta}} P(\mathbf{x} \mid \bar{\theta}, \bar{\beta}) b(\bar{\theta}, \bar{\beta}) d\bar{\theta} d\bar{\beta}} \ . \tag{7.13}$$

Given $b'(\theta, \beta)$, we now can use any of (7.6), (7.7) or (7.8). Next, we discuss making inference with (7.12) and (7.13) tractable.

7.3.2 Approximation

Although the proposed formalism enables us to capture if the robot's representation cannot explain the human's input, it is non-trivial to implement tractably for continuous β and θ, and large state and action spaces. Notice that equations (7.12) and (7.13) constitute a doubly-intractable system with denominators that cannot be computed exactly. For this reason, we employ several approximations to demonstrate the benefits of estimating situational confidence. Note that we do not consider these a contribution of our work: we choose the simplest approximations that facilitate tractability. There are many methods for approximate inference of θ studied in the literature that could be used for the joint (θ, β) spaces as well, from Metropolis Hastings [123, 250], to acquiring an MLE only via importance sampling of the partition function [96] or via a Laplace approximation [176].

To approximate the intractable integral in (7.12), we sampled a set $\mathcal{X}$ of 1500 trajectories. We sampled rewards according to (7.11) given by random unit norm θs, then optimized them with an off-the-shelf trajectory optimizer. We used TrajOpt [255], which is based on sequential quadratic programming and uses convex-convex collision checking. This way, we obtain dynamically feasible trajectories that optimize for different features in varying proportions. While this sampling strategy cannot be justified theoretically, it works well

[2]For deterministic (7.1), $P(\mathbf{u}_H \mid x^0, \mathbf{u}_R; \theta, \beta)$ is equivalent to $P(\mathbf{x} \mid \theta, \beta)$.

in practice: the resulting optimized trajectories are a heuristic for sampling diverse and interesting trajectories in the environment. Future work will address this shortcoming by either providing theoretical guarantees or using importance sampling instead.

For the second approximation to (7.13), we discretized the space of $\theta \in \Theta$ and $\beta \in \mathcal{B}$ into sets Θ_D and $\mathcal{B}_D$, which leaves us with a finite, easy to compute posterior. For more practical details on specific discretization schemes, see Sec. 7.8.1.

Using the above discretization[3], we can now perform tractable inference from demonstrations $\mathcal{D}$ to obtain a discrete posterior $b(\theta, \beta)$. Algorithm 4 summarizes the full procedure: given $\Theta_D, \mathcal{B}_D, \mathcal{X}$, and $\mathcal{D}$, our method iteratively updates the belief using (7.12) and (7.13), resulting in the posterior $b(\theta, \beta)$. Lacking any a-priori information, we chose a uniform prior but our method will work with any prior. We next present examples for what this posterior looks like in different scenarios.

Algorithm 4: Learning from Demonstrations (Offline)

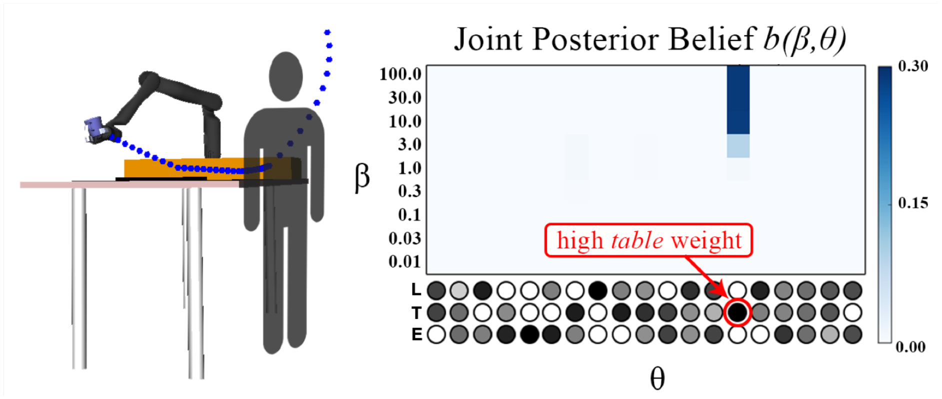
```
Input: Discretized sets $\Theta_D, \mathcal{B}_D, \mathcal{X}$, set of demonstrations $\mathcal{D}$.
$b(\theta, \beta) \leftarrow Uniform(\theta, \beta)$.
for x in $\mathcal{D}$ do
    for $\theta \in \Theta_D, \beta \in \mathcal{B}_D$ do
        $P(\mathbf{x} \mid \theta, \beta) = \frac{e^{\beta\theta^T \Phi(\mathbf{x})}}{\sum_{\bar{\mathbf{x}} \in \mathcal{X}} e^{\beta\theta^T \Phi(\bar{\mathbf{x}})}}$ as per (7.12).
        $b(\theta, \beta) \leftarrow \frac{P(\mathbf{x}|\theta,\beta) b(\theta,\beta)}{\sum_{\bar{\theta} \in \Theta, \bar{\beta} \in \mathcal{B}} P(\mathbf{x}|\bar{\theta},\bar{\beta}) b(\bar{\theta},\bar{\beta})}$ as per (7.13).
    end
end
return Posterior belief $b(\theta, \beta)$ inferred from $\mathcal{D}$.
```

7.3.3 Examples

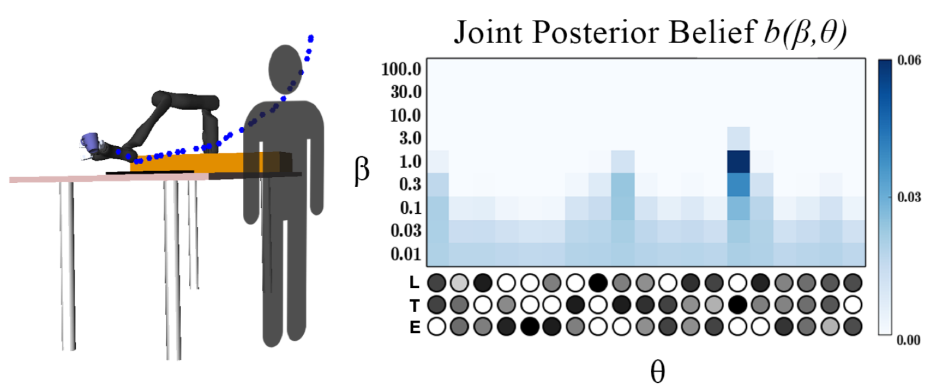
To provide intuition for how situational confidence can indicate when a robot's representation is misaligned, we illustrate some examples with a robot manipulator learning from a human demonstrator. These examples help prepare the setup we will present in our actual experiments in Sec. 7.5.

The robot manipulator is performing a household task of moving cups from a shelf onto the kitchen table. The robot needs to learn from the person's demonstrations how to best perform this task. For this purpose, the person physically guides the robot through

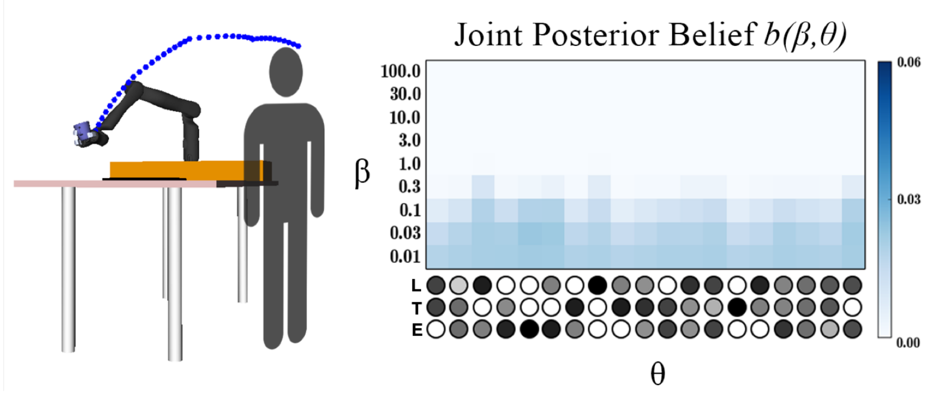
[3]In situations where the designer might want high fidelity inference over a large space of θ vectors, reasoning over a heavily discretized space would be more computationally expensive. However, longer offline computation is possible in our learning-from-demonstrations scenario as the inference happens offline, after providing the robot with human demonstrations. Alternatively, we could use Monte Carlo sampling approaches, similar to [123, 233].



(a) (Left) Simulated perfect demonstration for keeping the cup close to the table. (Right) Posterior belief resulted from this demonstration. A perfect demonstration leads to a high probability on the correct θ and high values for β.



(b) (Left) Noisy human demonstration for keeping the cup close to the table. (Right) Posterior belief resulted from this demonstration. A noisy but well-explained demonstration leads to a high probability on the correct θ and moderately high values for β. However, the noise in the demonstration reduces the probability at the distributional peak.



(c) (Left) Simulated perfect demonstration for keeping the cup away from the human's body. (Right) Posterior belief resulted from this demonstration. Notice that, since this demonstration is poorly explained (the robot is not reasoning about distance from the human), the posterior belief is spread out approximately uniformly over all θs and the lowest β values. This indicates that the robot cannot tell what the demonstration was intended for.

Figure 7.3: Three examples of demonstrations and the inferred posterior belief after each one of them. The robot infers the right $\theta = [0, 1, 0]$ from the two well-explained demonstrations, but, unlike the perfect simulated demonstration in Fig. 7.3a, the noisy one in Fig. 7.3b cannot reach the highest β and has as overall more spread-out probability distribution with a lower peak value. Lastly, the perfect simulated demonstration that is poorly explained in Fig. 7.3c results in a posterior that is spread-out over all θs and the lowest βs , consistent with the robot not being able to tell what the human's objective was.

one or a few demonstrations of moving the cup down to the table, from which the robot infers the hidden objective function.

In these examples, the robot's representation includes three features: efficiency (E) as sum of squared velocities over the trajectory, distance to the table (T), and distance from the laptop (L) depicted in black. Formally, we can represent these three features as:

$$\Phi(\mathbf{x}) = \begin{bmatrix} \sum_{i=1}^{T}((x^i - x^{i-1})/\Delta t)^2 \\ \sum_{i=0}^{T} ||x^i - x_{\text{table}}||_2 \\ \sum_{i=0}^{T} \max\{0, L - ||x^i - x_{\text{laptop}}||_2\} \end{bmatrix} \tag{7.14}$$

where L is the radius of a penalty sphere around the laptop, Δt is the discrete timestep between the states in the trajectory, and the corresponding feature weight vector is $\theta \in \mathbb{R}^3$.

Fig. 7.3 demonstrates how the feature weight θ and the situational confidence β are affected for well-explained, noisy, and poorly-explained simulated human demonstration. The posterior belief is shown for the combination of discrete parameters θ and β. Higher β indicates higher situational confidence. The three circles under each column represent the θ vector for that column, with the components being the efficiency, distance from the table, and distance from the laptop features. A larger feature weight is indicated by a darker circle, while a white color indicates zero weight. We used a cost function implementation, so here a larger weight indicates a larger penalty on that feature (or a smaller reward).

First, in Fig. 7.3a, we consider the case where the demonstration is a perfectly optimal trajectory produced by TrajOpt [255]. This serves as a sanity check for when the human and the robot have the same representation and the demonstration is perfect. The optimal demonstration was produced by finding a trajectory that moves the cup from the start configuration to the end while minimizing the distance between the cup and the table. Notice that with a perfect demonstration, the posterior distribution places the most probability mass on the θ that indicates high penalties for staying away from the table but no penalties for lack of efficiency or closeness to laptop. Moreover, the posterior also reveals that the most likely θ also corresponds *with the highest available confidence* β.

Next, in Fig. 7.3b we recorded a real human demonstration of the same cup-to-table behavior. The nature of demonstrations both on hardware and from real people introduce noise into the demonstration, making it potentially suboptimal with respect to the robot's model. However, in this case the human and the robot still share the same representation (i.e. the robot and the human both know about the the efficiency, table, and laptop features). Here, we study how the noise in the demonstration affects the robot's inference. Notice that even with an imperfect demonstration, the robot is able to identify the correct θ parameter, but now with a lower confidence β.

Lastly, we consider the example where the demonstration is optimal but the robot does not have a rich enough representation to explain it. The robot reasons about the same three features, but now the demonstration was produced by optimizing for an additional feature that is outside its representation: keeping the cup away from the human's body. We observe that the probability distribution in Fig. 7.3c is spread over all the θ values in the space, with the highest values on low βs. This example shows how, in the case of poorly-explained input, the robot's inference is unsure which objective the human had in mind, and assigns low situational confidence to the given input.

These illustrative examples give us valuable insight into how the (θ, β)-belief changes depending on how well-explained the input is. For perfectly explained demonstrations, the inference identifies the correct θ with high posterior probability. As the input becomes more poorly-explained, the robot loses confidence in all θs, assigning approximately uniformly spread-out probability on the lowest situational confidence values β.

7.4 Approach: Misalignment for Corrections

7.4.1 Formulation

We consider the setting where human input is given as physical interventions during the robot's task execution. Fig. 7.2 (right) is an example of such a correction. The human may provide a correction to improve some aspect of the task execution that is not represented in the robot's objective space. When the robot receives input, it should be able to reason about its situational confidence in light of the correction and replan its trajectory accordingly for the rest of the task execution or until a new correction happens. Thus, the robot must have access to an inference algorithm that can run in real time. In this section, we will present an online version of our situational confidence framework.

In the physical corrections setting, the robot starts with an initial guess of the parameter θ and uses a trajectory optimization scheme to compute a motion plan seeking to minimize the associated reward R_θ. The robot performs the task at hand by applying controls $\mathbf{u}_R$ via an impedance controller in order to track the computed trajectory $\mathbf{x}$.

At any timestep t during the trajectory execution, the human may physically interact with the robot, inducing a joint torque u_H^t. When this happens, the robot can use the human input to update its estimated θ parameter, and thereby the corresponding objective R_θ. Given the new adapted objective, the robot replans an optimized trajectory $\mathbf{x}$ and tracks it until the next human input is sensed or until the task is completed.

Following [25], the robot's representation of the task assumes that the human does not explicitly care about the robot's control effort, but only about features of the state trajectory. The human is also assumed to have a preference for minimizing her own control effort. This captures the human's incentive to have the robot perform the task autonomously, providing only minimal input to guide the robot towards the correct behavior when necessary. Encompassing these assumptions, the reward (7.10) takes the form:

$$R_\theta(\mathbf{x}, u_H^t) = \theta^T \Phi(\mathbf{x}) - \lambda \|u_H^t\|^2. \tag{7.15}$$

To approximately compute the trajectory resulting from the human's input, we follow the approach in [25] and introduce the notion of a *deformed trajectory* $\mathbf{x}_D$. This trajectory constitutes the robot's estimate of the human's desired trajectory given her applied torque

u_H^t. Given the robot's default trajectory $\mathbf{x}_R := \mathbf{x}(\cdot; x^0, \mathbf{u}_R, \mathbf{0})$ and having observed the instantaneous human intervention u_H^t, we compute $\mathbf{x}_D$ by deforming the robot's default trajectory in the direction of u_H^t:

$$\mathbf{x}_D = \mathbf{x}_R + \mu A^{-1} \tilde{\mathbf{u}}_H \ , \tag{7.16}$$

where $\mu > 0$ scales the magnitude of the deformation, $A \in \mathbb{R}^{n(T+1) \times n(T+1)}$ defines a norm on the Hilbert space of trajectories[4] and dictates the deformation shape [88], and $\tilde{\mathbf{u}}_H \in \mathbb{R}^{n(T+1)}$ is u_H^t at indices nt through $n(t+1)$ and 0 otherwise. The human is therefore modeled by (7.15) as trading off between inducing a good trajectory $\mathbf{x}_D$ with respect to θ, and minimizing her effort.

Equipped with this reward function, we need the robot to reason about the reliability of its representation given new corrections. In contrast with our analysis in Sec. 7.3, here the person does not give full demonstrations $\mathbf{x}$, but instead offers corrections u_H^t based on the robot's default trajectory $\mathbf{x}_R$. Applying (7.4) to this setting, we have:

$$P(u_H^t \mid x^0, \mathbf{u}_R; \theta, \beta) = \frac{e^{\beta(\theta^\top \Phi(\mathbf{x}_D) - \lambda \|u_H^t\|^2)}}{\int e^{\beta(\theta^\top \Phi(\bar{\mathbf{x}}_D) - \lambda \|\bar{u}\|^2)} d\bar{u}} \ , \tag{7.17}$$

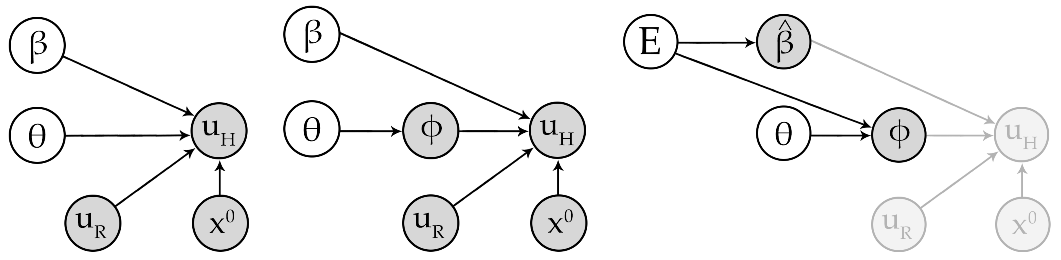
where $\mathbf{x}_D$ and $\bar{\mathbf{x}}_D$ are given by (7.16) applied to their respective controls u_H^t and $\bar{u}$.

Ideally, with this model of human actions, illustrated in Fig. 7.4a, we would perform inference over both the situational confidence β and the modeled parameters θ by maintaining a joint Bayesian belief $b'(\theta, \beta)$. Analogously to the demonstrations case, our probability distribution over θ would automatically adjust for well-explained corrections, whereas for poorly-explained ones the robot's posterior would not deviate significantly form its prior on θ. Unfortunately, this Bayesian update is not generally feasible in real time, given the continuous and possibly high-dimensional nature of the parameter space Θ. Even in simple scenarios with a small number of continuous features, discretizing Θ as we did in the demonstrations case would generally yield an overly slow inference, making the method impractical for use in the real-time collaborative scenarios that we are interested in here. Thus, to evaluate the benefits of estimating β we need to derive an online method that goes beyond simple discretization.

7.4.2 Approximation

To alleviate the computational challenge of performing joint inference over β and θ, we introduce a structural assumption that will enable us to approximately decouple the two inference problems.

[4]We used a norm A based on acceleration, consistent with [25], but other norms are possible as well.



(a) In the true graphical model, u_H is an observation of θ and the situational confidence β.

(b) We use the proxy variable Φ to first estimate β efficiently.

(c) We interpret the estimate $\hat{\beta}$ as an indirect observation of the unobserved E, which we then use for the θ estimate.

Figure 7.4: Graphical model formulation (a) and modifications to it ((b) and (c)) for real-time tractability.

7.4.2.1 Estimating β. To estimate β without dependence on θ, we assume that to decide what correction to provide, the human first chooses the desired features Φ of the resulting trajectory $\mathbf{x}_D$ and then selects an input u_H^t that will obtain these features (Fig. 7.4b).

Based on the observed human input u_H^t and the trajectory features of the deformed trajectory $\Phi(\mathbf{x}_D)$, the robot can obtain an estimate of β by considering how efficient the human's input was for the features achieved. Letting $\mathcal{U}_\Phi$ be the set of inputs that achieve the same observed features $\Phi_D := \Phi(\mathbf{x}_D)$, the Boltzmann decision model gives

$$P(u_H^t \mid x^0, \mathbf{u}_R; \Phi_D, \beta) = \frac{e^{\beta(\theta^\top \Phi_D - \lambda \|u_H^t\|^2)}}{\int_{\mathcal{U}_\Phi} e^{\beta(\theta^\top \Phi(\bar{\mathbf{x}}_D) - \lambda\|\bar{u}\|^2)} d\bar{u}} = \frac{e^{-\beta\lambda\|u_H^t\|^2}}{\int_{\mathcal{U}_\Phi} e^{-\beta\lambda\|\bar{u}\|^2} d\bar{u}}\ , \tag{7.18}$$

since the term $\theta^\top \Phi(\bar{\mathbf{x}}_D)$ is constant for all $\bar{u} \in \mathcal{U}_\Phi$ and equal $\theta^\top \Phi_D$ in the numerator.

Using (7.18), the robot can obtain an estimate of β by considering how efficient the human's correction was for the features achieved—if the input seems highly inefficient, this is indicative that the features modeled by the robot may not accurately capture the human's preference.

It is useful to approximate the integral over the constrained set $\mathcal{U}_\Phi \subset \mathcal{U}$ by an integral over the entire set of possible inputs $\mathcal{U}$, introducing a penalty term in the exponent that results in a soft indicator function for $\bar{u} \in \mathcal{U}_\Phi$:

$$P(u_H^t \mid x^0, \mathbf{u}_R; \Phi_D, \beta) \approx \frac{e^{-\beta\lambda\|u_H^t\|^2}}{\int_{\mathcal{U}} e^{-\beta(\lambda\|\bar{u}\|^2 + \kappa\|\Phi(\bar{\mathbf{x}}_D) - \Phi_D\|^2)} d\bar{u}}\ . \tag{7.19}$$

Note that for an arbitrarily large κ there is an arbitrarily small probability assigned to $\mathcal{U} \setminus \mathcal{U}_\Phi$ in the integral. It is now possible to apply the Laplace approximation to the unconstrained integral (see Sec. 7.8.3 for details), yielding:

$$P(u_H^t \mid x^0, \mathbf{u}_R; \Phi_D, \beta) \approx \frac{e^{-\beta\lambda\|u_H^t\|^2}}{e^{-\beta(\lambda\|u_H^*\|^2 + \kappa\|\Phi(\mathbf{x}_D^*) - \Phi_D\|^2)}} \sqrt{\frac{\beta^k |H_{u_H^*}|}{2\pi^k}}\ , \tag{7.20}$$

where k is the action space dimensionality and $H_{u_H^*}$ is the Hessian of the exponent in the denominator of (7.19) around u_H^*. We obtain the optimal action u_H^* by solving the constrained optimization problem (see Sec. 7.8.2):

$$\begin{aligned} \underset{\tilde{u}_H}{\text{minimize}} \quad & \|\tilde{u}_H\|^2 \\ \text{subject to} \quad & \Phi(\mathbf{x} + \mu A^{-1}\tilde{\mathbf{u}}_H) - \Phi_D = 0 \ . \end{aligned} \tag{7.21}$$

In other words, the resulting u_H^* is the minimal norm $\tilde{u}_H$ the human could have taken, constrained to lie in $\mathcal{U}_\Phi$. As such, the second norm in the denominator's exponent is 0, and the final conditional probability becomes:

$$P(u_H^t \mid x^0, \mathbf{u}_R; \Phi_D, \beta) = e^{-\beta\lambda(\|u_H^t\|^2 - \|u_H^*\|^2)} \sqrt{\frac{\beta^k |H_{u_H^*}|}{2\pi^k}} \ . \tag{7.22}$$

We derive below the maximum likelihood estimator (MLE), noting that a maximum *a posteriori* (MAP) estimator is often appropriate given a certain prior on β.

$$\begin{aligned} \hat{\beta} &= \arg\max_{\beta} \{\log(P(u_H^t \mid x^0, \mathbf{u}_R; \Phi_D, \beta)\} \\ &= \arg\max_{\beta} \{-\beta\lambda(\|u_H^t\|^2 - \|u_H^*\|^2) + \log(\sqrt{\frac{\beta^k |H_{u_H^*}|}{2\pi^k}})\}. \end{aligned} \tag{7.23}$$

Applying the first-order condition and setting the derivative to zero yields the maximizer:

$$\hat{\beta} = \frac{k}{2\lambda(\|u_H^t\|^2 - \|u_H^*\|^2)} \ . \tag{7.24}$$

The estimator[5] above yields a high value when the difference between u_H^t and u_H^* is small, i.e. the person's correction achieves the induced features $\Phi(\mathbf{x}_D)$ efficiently. If $\mathbf{x}_D$ brings the robot closer to the table, and u_H^t pushes the robot straight towards the table, u_H^t is an efficient way to induce those new feature values. However, when there is a much more efficient alternative (e.g. when the person pushes mostly sideways rather than straight towards the table), $\hat{\beta}$ will be small. Efficient ways to induce the feature values will suggest well-explained inputs, inefficient ones will suggest poorly-explained corrections.

7.4.2.2 Estimating θ. To tractably estimate θ building on the β estimate, we introduce an auxiliary binary variable $E \in \{0, 1\}$ indicating whether the human's intervention can be well *explained* by the robot's modeled reward features. We will perform offline training with ground-truth access to this variable to learn its relation to the robot's estimate $\hat{\beta}$.

[5]Note that $\hat{\beta}$ is non-negative, since u_H^* is the minimal-norm $\tilde{u}_H$ that satisfies the constraint, so the difference in the denominator of (7.24) is positive.

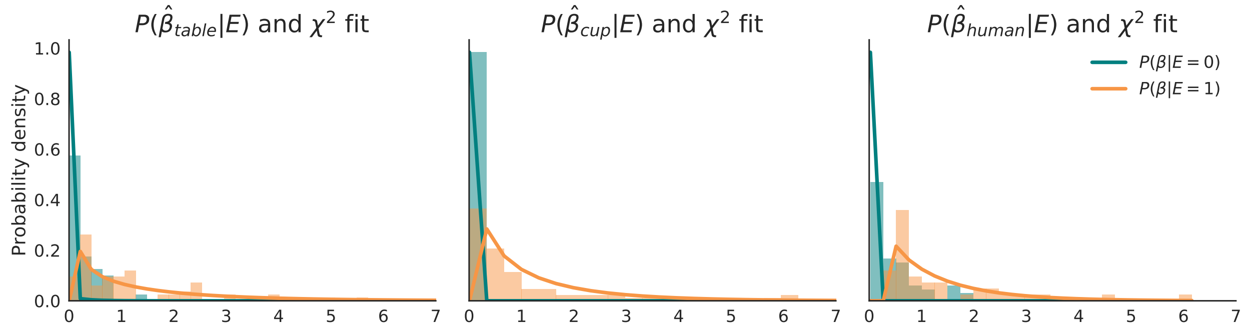


Figure 7.5: Empirical estimates for $P(\hat{\beta} \mid E)$ and their corresponding chi-squared (χ^2) fits.

When $E = 1$, the human's desired modification of the robot's behavior can be well explained by *some* vector $\theta \in \Theta$, which will lead the intervention to appear less noisy to the robot (i.e. β is large). As a result, the correction u_H^t is likely to be efficient for the reward encoded by this θ. Conversely, when $E = 0$, the intervention appears noisy (i.e. β is small), and the human's correction cannot be well explained by any of the reward features modeled by the robot.

The graphical model in Fig. 7.4c relates the induced feature values Φ_D to θ as a function of the E. When $E = 1$, the induced features will tend to have high reward with respect to θ; when $E = 0$, the induced features *do not depend on* θ, and we model them as Gaussian noise centered around the feature values of the robot's currently planned trajectory $\mathbf{x}_R$.

$$P(\Phi_D \mid \theta, E) = \begin{cases} \dfrac{e^{\theta^\top \Phi_D}}{\int e^{\theta^\top \Phi(\tilde{\mathbf{x}}_D)} d\tilde{\mathbf{x}}_D}, & E = 1 \\ \left(\frac{\nu}{\pi}\right)^{\frac{k}{2}} e^{-\nu ||\Phi_D - \Phi(\mathbf{x}_R)||^2}, & E = 0 \end{cases} \tag{7.25}$$

with the constant in the $E = 0$ case corresponding to the Gaussian normalization term.

In addition, this graphical model relates the $\hat{\beta}$ resulting from the model in Fig. 7.4b to E by a $P(\hat{\beta} \mid E)$. We fit this distribution from controlled user interaction samples where we have ground-truth knowledge of E[6]. For each sample interaction, we compute $\hat{\beta}$ (for example, using (7.24) if using MLE) and label it with the corresponding binary E value. We fit a chi-squared distribution to these samples to obtain the probability distributions for $P(\hat{\beta} \mid E = 0)$ and $P(\hat{\beta} \mid E = 1)$. The resulting distributions are shown in Fig. 7.5[7].

Using the model in Fig. 7.4c with the learned distribution $P(\hat{\beta} \mid E)$, we can infer a θ estimate in real time whenever a physical correction from the human is measured. We do

[6]Since we tell users what to optimize for, we know whether the human's input is well-explained with respect to the robot's representation or not.

[7]Because users tend to accidentally correct more than one feature, we perform β-inference separately for each feature. This requires more overall computation (although still linear in the number of features, and can be parallelized) and a separate $P(\hat{\beta} \mid E)$ estimate for each feature.

this tractably by interpreting the $\hat{\beta}$ obtained from (7.24) as an indirect observation of the unknown E. We combine the empirically characterized likelihood model $P(\hat{\beta} \mid E)$ with an initial uniform prior $P(E)$ to maintain a Bayesian posterior on E based on the evidence $\hat{\beta}$ constructed from human observations at deployment time, $P(E \mid \hat{\beta}) \propto P(\hat{\beta} \mid E)P(E)$.

Further, since we wish to obtain a posterior estimate of the human's objective θ, we use the model from Fig. 7.4c to obtain the posterior probability measure

$$P(\theta \mid \Phi_D, \hat{\beta}) \propto \sum_{E\in\{0,1\}} P\big(\Phi_D \mid \theta, E\big)P(E \mid \hat{\beta})P(\theta) \ . \tag{7.26}$$

Following [25], we note that we can approximate the partition function in the human's policy (7.25) by employing the Laplace approximation. Taking a second-order Taylor series expansion of the exponent's objective about $\mathbf{x}_R$, the robot's current best guess at the optimal trajectory, we obtain a Gaussian integral that can be evaluated in closed form

$$P(\Phi_D \mid \theta, E = 1) \approx e^{\theta^\top \big(\Phi_D - \Phi(\mathbf{x}_R)\big)} \ . \tag{7.27}$$

We also use a Gaussian prior distribution of θ around the robot's current estimate $\hat{\theta}$:

$$P(\theta) = \frac{1}{(2\pi\alpha)^{\frac{k}{2}}} e^{-\frac{1}{2\alpha}||\theta-\hat{\theta}||^2} \ , \tag{7.28}$$

where $\alpha \geq 0$ determines the variance of the Gaussian.

To obtain an update rule for the θ parameter, we can simply plug (7.25), (7.27), and (7.28) into (7.26). For legibility, let's denote $\Gamma(\Phi_D, E = i) = P(E = i \mid \hat{\beta})P\big(\Phi_D \mid \theta, E = i\big)$, for $i \in \{0, 1\}$. Then, the maximum-a-posteriori estimate of the human's objective θ is the solution maximizer of

$$\begin{aligned} &P(\theta)\Big[\Gamma(\Phi_D, E = 1) + \Gamma(\Phi_D, E = 0)\Big] \\ &= \frac{1}{(2\pi\alpha)^{\frac{k}{2}}} e^{-\frac{1}{2\alpha}||\theta-\hat{\theta}||^2}\Big[P(E = 1 \mid \hat{\beta})e^{\theta^\top \big(\Phi_D - \Phi(\mathbf{x}_R)\big)} + P(E = 0 \mid \hat{\beta}) \left(\frac{\nu}{\pi}\right)^{\frac{k}{2}} e^{-\nu||\Phi_D - \Phi(\mathbf{x}_R)||^2}\Big] \ . \end{aligned} \tag{7.29}$$

Setting the derivative of (7.29) w.r.t. θ to 0 gives the maximum-a-posteriori update

$$\hat{\theta}' = \hat{\theta} + \alpha \frac{\Gamma(\Phi_D, E = 1)}{\Gamma(\Phi_D, E = 1) + \Gamma(\Phi_D, E = 0)} \big(\Phi_D - \Phi(\mathbf{x}_R)\big) \ . \tag{7.30}$$

We note that due to the coupling in $\hat{\theta}'$, the solution to (7.30) is non-analytic and can instead be obtained via numerical approaches like Newton-Raphson or quasi-Newton methods.

In previous objective-learning approaches including [25] and [311], it is implicitly assumed that all human actions are fully explainable by the robot's representation of the objective function space ($E = 1$), leading to the simplified update

$$\hat{\theta}' = \hat{\theta} + \alpha\big(\Phi_D - \Phi(\mathbf{x}_R)\big) \ , \tag{7.31}$$

which can be easily seen to be a special case of (7.30) when $P(E = 0 \mid \hat{\beta}) \equiv 0$. Our proposed update rule therefore *generalizes* commonly-used objective-learning formulations to cases where the human's underlying objective function is not fully captured by the robot's model. We expect that this extended formulation will enable learning that is more robust to misaligned or incomplete human objective parameterizations.[8] Once we obtain the $\hat{\theta}'$ update, we replan the robot trajectory in its 7-DOF configuration space with an off-the-shelf trajectory optimizer, TrajOpt [255].

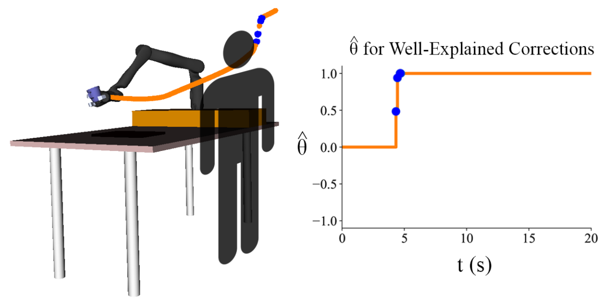
The update rule changes the weights in the objective in the direction of the feature difference as well, but how much it does so depends on the probability assigned to the correction being well-explained. Looking back at Sec. 7.2, this update is approximating (7.7). At one extreme, if we know with full certainty that the correction is well explained, then we do the full update as in traditional objective learning. But crucially, at the other extreme, if we know that the correction is poorly explained, we do not update at all and keep our prior belief.

Algorithm 5: Learning from Corrections (Online)

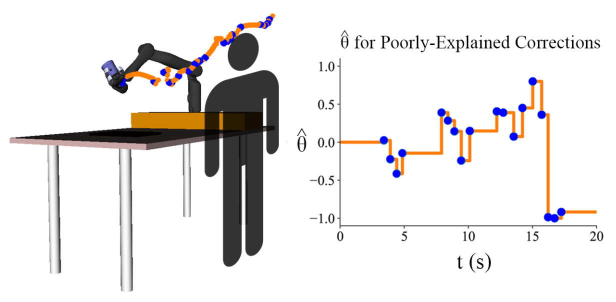
```
**Input:** $P(\hat{\beta} \mid E = i), \forall i \in \{0, 1\}$ from training data.
Initialize $\mathbf{x}_R \leftarrow TrajOpt(\hat{\theta})$ for initial $\hat{\theta}$.
**while** *goal not reached* **do**
    **if** $u_H \neq \boldsymbol{0}$ **then**
        $\mathbf{x}_D = \mathbf{x}_R + \mu A^{-1}\tilde{\mathbf{u}}_H$ .
        $u_H^* \leftarrow OptimalHumanAction(\Phi_D)$, as per (7.21) .
        $\hat{\beta} = \frac{k}{2\lambda(\|u_H\|^2 - \|u_H^*\|^2)}$, as per (7.24) .
        $\hat{\theta} \leftarrow \hat{\theta} + \alpha \frac{\Gamma(\Phi_D, E=1)}{\Gamma(\Phi_D, E=1) + \Gamma(\Phi_D, E=0)} \big(\Phi_D - \Phi(\mathbf{x}_R)\big)$, as per (7.30) .
        $\mathbf{x}_R \leftarrow TrajOpt(\hat{\theta})$ .
    **end**
**end**
```

The full algorithm is given in Algorithm 5. The robot begins tracking a trajectory $\mathbf{x}_R$ given by an initial $\hat{\theta}$. Once a human torque u_H is sensed, the robot deforms its trajectory to compute the induced features Φ_D, computes the optimal human action u_H^* using (7.21), and uses it to estimate $\hat{\beta}$ for that input. It then updates $\hat{\theta}$ using the learned distributions $P(\hat{\beta} \mid E = i), \forall i \in \{0, 1\}$, and updates its tracked trajectory $\mathbf{x}_R$. For more practical details on how replanning works, and how to set various hyperparameters, consult Sec. 7.8.2.

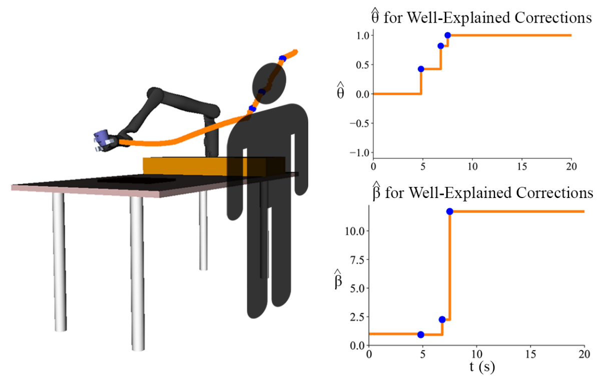
[8]Note that to enforce the constraint on $||\theta|| = 1$, we can indeed project the resulting $\hat{\theta}'$ onto the unit ball. In practice, because our learning from corrections algorithm separates the β-inference from the θ-inference, this projection is no longer required, but we found it helpful to still constrain the space of Θ to encourage smoothness in the change of the reward function.



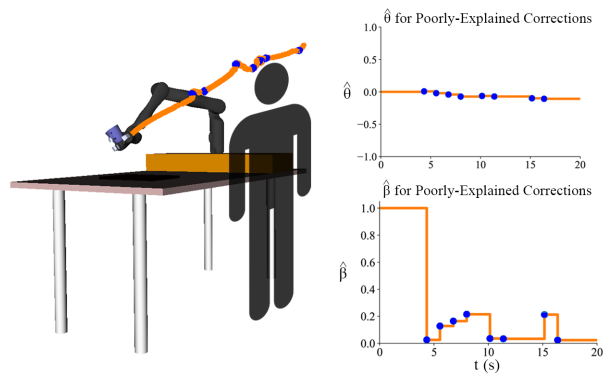
(a) (Left) The human applies well-explained corrections to keep the cup close to the table. Learning with fixed β leads to a correct trajectory that satisfies the human's preference. (Right) As the person corrects the robot by pushing down on it, the learning algorithm gradually updates its weight on the distance to table feature.



(b) (Left) The human applies poorly-explained corrections to keep the cup upright. Learning with fixed β leads to a oscillatory and noisy trajectory. (Right) The naive learning algorithm incorrectly updates the weight on the distance to table feature, leading to unintended learning.



(c) (Left) The human applies well-explained corrections to keep the cup close to the table. Learning with estimated β leads to a correct trajectory that satisfies the human's preference. (Right) As the person corrects the robot by pushing down on it, the learning algorithm infers high $\hat{\beta}$ and updates its weight on the distance to table feature.



(d) (Left) The human applies poorly-explained corrections to keep the cup upright. Learning with estimated β leads to a smooth trajectory where the robot is robust to poorly-explained inputs. (Right) The confidence-aware learning algorithm infers low $\hat{\beta}$ and correctly avoids unintended learning for the distance to table feature.

Figure 7.6: Examples of physical corrections (blue interaction points) and the resulting behavior for the fixed β method (top) and estimated β method (bottom). When the corrections are well explained, both methods learn the correct weight $\hat{\theta} = 1.0$. But if they are poorly-explained, our method infers low $\hat{\beta}$ and manages to reduce unintended learning, whereas the fixed β method produces incorrect oscillatory behavior.

7.4.3 Examples

As in Sec. 7.3, we now illustrate some examples to help lay out some of the setup we will present in our actual experiments in Sec. 7.5 and Sec. 7.6. We provide intuition for how the estimators of β and θ work when we have a perfectly aligned representation and a misaligned representation. In all of the examples, the robot reasons about the previously described distance from the table feature. What changes is the feature for which the human decides to provide corrections.

We look at two situations: the human may correct the relevant feature and push the robot closer to the table, or she might provide an poorly-explained input to keep the coffee mug upright. Fig. 7.6 illustrates the two scenarios and contrasts our estimated-β approach to the state of the art fixed-β approach that uses (7.31).

On the top we present the fixed-β method and its performance with both the well-explained and the poorly-explained input. When the input is well explained, the left image shows that the robot learns from the interactions and converges close to the true $\theta = 1$. However, when the input is poorly explained on the right, the robot incorrectly learns fictitious θ values and produces oscillatory behavior.

In Fig. 7.6 (bottom) we showcase our described estimated-β method. If inputs are well-explained, the value for $\hat{\beta}$ increases, allowing θ to grow up to the real value $\theta = 1$. The method has the same behavior as the fixed-β method. However, more importantly, if inputs are poorly-explained, our method immediately estimates low $\hat{\beta}$ values, which allows it to significantly reduce unintended learning as compared to the fixed-β method.

This figure illustrates how situational confidence estimation can aid the robot when the human input is poorly explained. We stress that although our method does not allow the robot to magically learn the correct behavior that the user desires, it greatly reduces unintended learning and undesired behaviors.

7.5 Experiments: Real Robot Case Studies

Equipped with our algorithmic approaches to situational confidence estimation, we now consider two case studies in learning from demonstrations and corrections using real human input on a 7-DoF robot manipulator.

7.5.1 Demonstrations

We collected human demonstrations of household motion planning tasks and performed our situational confidence inference offline. We recruited 12 people to physically interact with a JACO 7-DoF robotic arm and analyzed 4 common cases that can arise in the context of personal robotics learning.

For all the experiments in this section, we asked the participants to provide demonstrations with respect to a feature of interest, which the robot might (well-explained) or might not (poorly-explained) have in its representation. Some of the features that the humans had to prioritize include: distance of the end effector from the table, distance from the person, or distance from the end-effector to a laptop placed on the table.

Before giving any demonstrations, each person was allowed a period of training with the robot in gravity compensation mode to get accustomed to interacting with the robot. When collecting demonstrations, participants were asked to move the robot arm holding a cup of coffee from the upper shelf of a cupboard to right above the table, across a laptop.

After collecting all demonstrations, we designed the robot's representation for inference. In all scenarios the robot reasons over the same three features as in (7.14): E, T, and

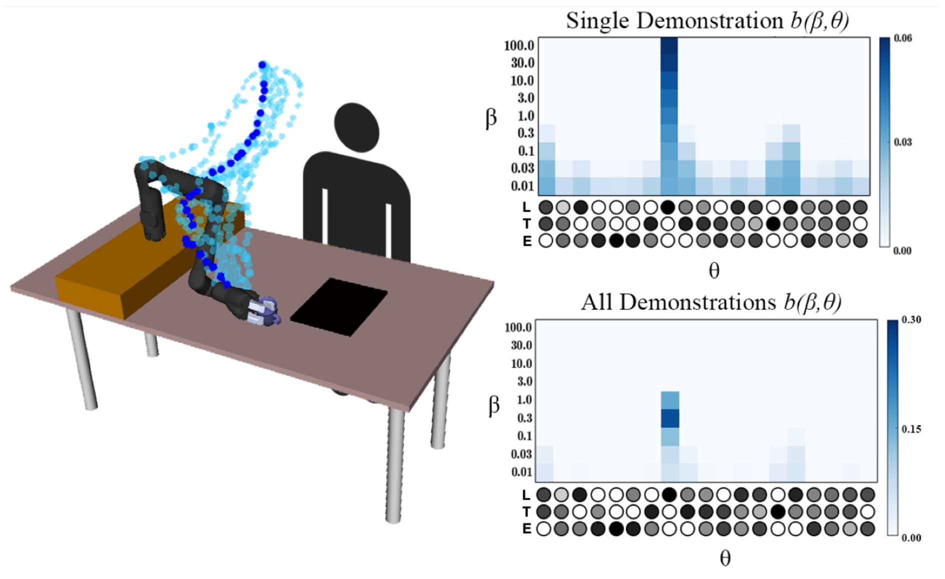


Figure 7.7: (Left) Human demonstrations avoiding the laptop. (Right) Upper distribution is the posterior belief for the highlighted blue demonstration. Since the robot has the laptop feature in its representation, this demonstration induces a high β on the correct $\theta = [0, 0, 1]$. Below, when considering all the demonstrations, the inference procedure converges to a slightly lower β value due to the suboptimality of some of the demonstrations in the dataset.

L. Although the robot always knows about these features, the demonstrations may have been given relative to different (and potentially unknown) features.

Throughout our scenarios, we tested two hypotheses:

H1. *If the human input is well-explained, our inference procedure places high probability on the correct θ hypothesis, with a high situational confidence β.*

H2. *If the human input is poorly-explained, our inference procedure does not place high probability on any θ hypothesis and is uniform over all hypotheses with low situational confidence β.*

To test these hypotheses, we looked at the resulting inferred belief. Given the demonstrations and a parametrization of the reward function, we first updated the belief over the weight and situational confidence parameters for each demonstration, $b_{single}(\theta, \beta)$. This gives insights into how a single demonstration can affect the robot's inference procedure.

Next, we used all 12 human demonstrations to obtain a probability distribution over the weight and confidence, $b_{all}(\theta, \beta)$, for each scenario. By using multiple demonstrations as evidence about the reward and the situational confidence parameter, we see how in some scenarios multiple demonstrations can help improve confidence in the θ estimation.

7.5.1.1 Well-aligned Representation. Here we consider a scenario where the robot and the human share the same representation, i.e. the robot's model is well-aligned. The participants were instructed to avoid spilling the coffee over the laptop by providing a demonstration where the robot's end-effector is away from the electronic device. Here, the

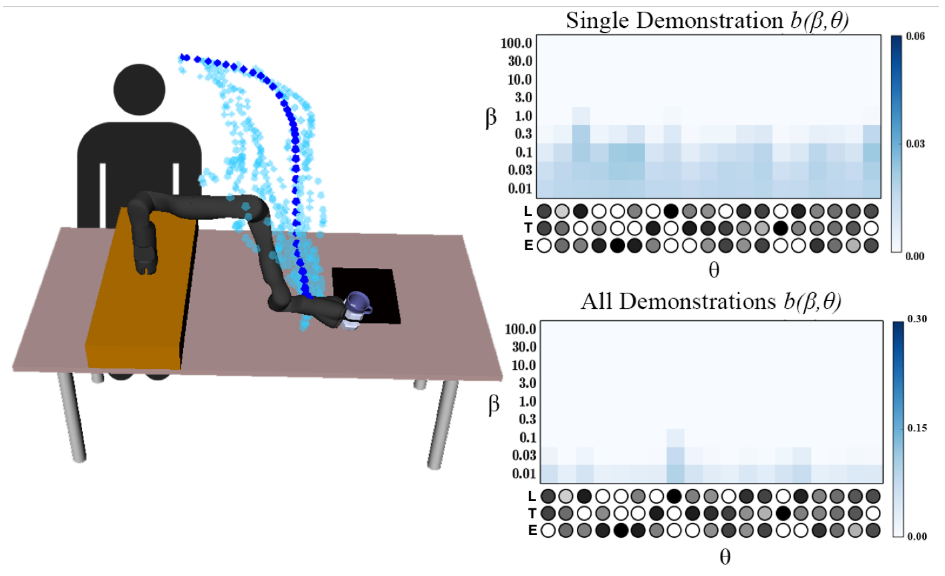


Figure 7.8: (Left) Human demonstrations avoiding the user's body. (Right) Upper distribution is the posterior belief $b(\beta, \theta)$ for the highlighted demonstration. Since the robot's model does not include distance to the user's body, none of the robot's hypotheses can explain the demonstration, as reflected in the higher probabilities on low βs for all θs. After performing inference on all the demonstrations, the distribution in the lower right plot shows even more probability mass on the lowest situational confidence values.

feature of interest was the distance from the laptop which was in the robot's representation: the demonstration would be well explained as long as the demonstration maintained a distance of at least L meters away from the center of the laptop.

On the left of Fig. 7.7 we visualize all 12 recorded demonstrations and the experimental setup. Note that most participants had an easy time providing demonstrations which avoided the laptop. Indeed, we noticed that 10 out of the 12 demonstrations resulted in high situational confidence and a probability distribution similar to the one at the top right of Fig. 7.7. Here, the θ vector that has largest weight on the third feature (distance from the laptop) is correctly inferred to have high β value. This signals that the robot is highly confident the person provided a demonstration that avoids the laptop, which is correct and supports our hypothesis H1.

Another interesting observation is that the situational confidence over all 12 demonstrations together is lower than in the case of the single optimal demonstration highlighted in blue (peak at around 1.0 instead of 100.0)[9]. This is due to the two noisy demonstrations that came too close to the laptop. When working with non-expert users, it is inevitable that such imperfect demonstrations will arise. However, despite the challenge of noisy

[9]In the lower right belief in Fig. 7.7, note from the colorbar values that the probability mass is more peaked than in the case of a single demonstration. This confirms our intuition that the robot's certainty in the hypothesis is enhanced the more demonstrations supporting that hypothesis it receives.

and/or erroneous demonstrations, the algorithm recovers the correct θ hypothesis with a relatively high β, supporting H1 once again.

7.5.1.2 Misaligned Representation. We look at the opposite scenario: the robot and the human do not share the same representation and the robot's model is clearly misaligned.

Participants were instructed to move the robot from start to end while also keeping the robot's hand away from their body to avoid spilling coffee on their clothes. Since the robot's reward does not include distance to humans, the demonstrations should appear poorly explained relative to the robot's model of how humans choose demonstrations.

Fig. 7.8 visualizes all 12 demonstrations as well as the posterior probability distributions for a single highlighted trajectory and for all 12. For both a single demonstration and all of them, in the case of misalignment none of the hypotheses are correct. Thus, the robot infers equally low probability for all θs, with low situational confidence, supporting our hypothesis H2. This signals that the robot is unsure what the person's demonstration referred to, as we expected.

These two examples illustrate cases where our method supports the two hypotheses above. However, there are important limitations that we discuss next.

7.5.1.3 Feature Correlation. The past two examples demonstrate clear instances when the robot's objective space is either well- or mis-aligned. However, often times situations will be more ambiguous. For example, although the human input may refer to a feature that is nonexistent in the robot's representation, the robot may know about a feature that is correlated to it. In this next scenario, we investigate how such feature correlation influences the situational confidence estimates.

We asked the participants to move the robot from the same start and end as before, while keeping the cup in the robot's end-effector away from their body to avoid spilling coffee on their clothes. The setup is similar to the poorly-explained demonstration in the previous scenario, only that now the human starts in a different initial position.

Visualizations of the 12 demonstrations in Fig. 7.9 showcase that although all demonstrations move the cup away from the person, some of them (depicted in blue) also maintain a good distance away from the laptop. Hence, even though the human was trying to teach the robot to stay away from their body, the robot interprets the human's demonstrations as a signal to stay away from the laptop. Thus, we say that the distance from human and distance from laptop features are *correlated*.

When looking at the top-right posterior probability in Fig. 7.9, the distribution over θ, β shows that our algorithm infers a high situational confidence for the θ that fully considers the distance from the laptop. Thus, even if the human input does not pertain to a feature in the robot's representation, in some cases the demonstration can still be explained via correlated features in the robot's representation. This observation does not support H2 and is clearly a limitation of our method.

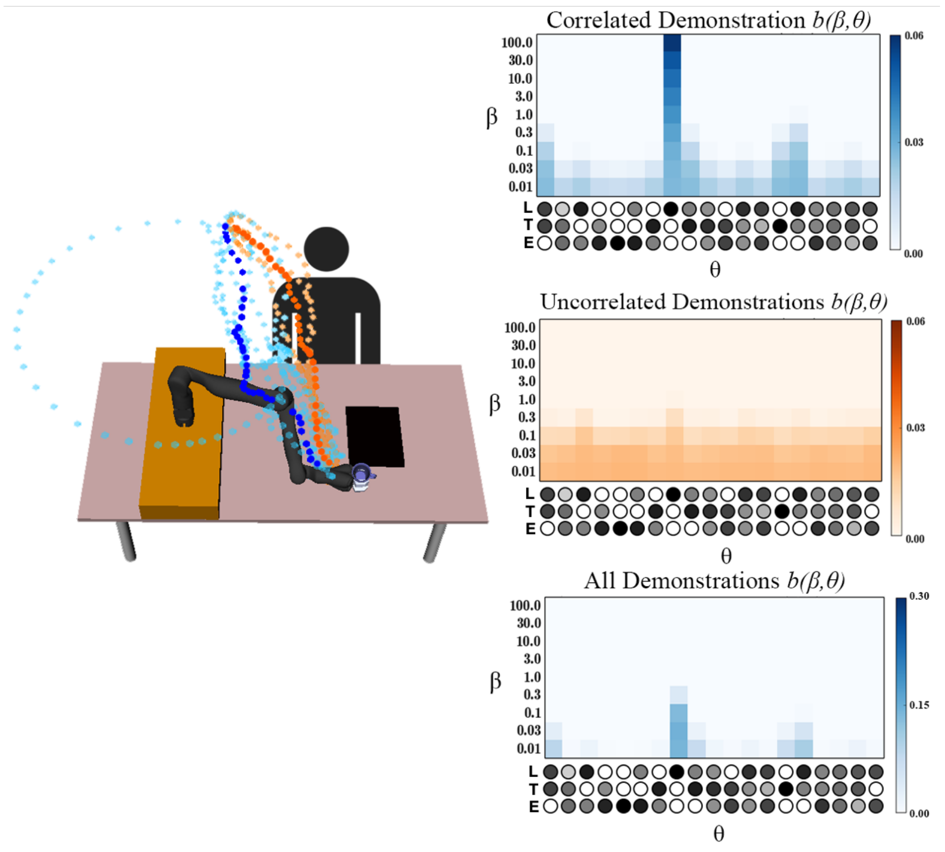


Figure 7.9: (Left) Human demonstrations avoiding the user's body. The blue cluster is correlated with the feature describing distance from the laptop. The orange cluster is uncorrelated. (Right) The top distribution is the posterior belief $b(\beta, \theta)$ for the highlighted blue correlated demonstration. Notice that the hypothesis that puts all weight on avoiding the laptop $\theta = [0, 0, 1]$ dominates the distribution. Meanwhile, the posterior belief for the highlighted orange demonstration indicates low situational confidence in all hypotheses. The bottom distribution shows that when combining all demonstrations, the robot continues to have low situational confidence although the laptop hypothesis has slightly higher β.

However, the orange cluster of demonstrations in Fig. 7.9, showcase the fine line between demonstrations that induce a feature correlation and those that do not. The orange demonstrations clearly ignore the laptop and simply take the shortest path to the end goal while avoiding the human's body. As we can see in the orange probability distribution, our method infers a uniform distribution over all θ hypotheses, with a focus on the lowest situational confidence values, backing H2.

These two clusters highlight that our method infers reasonable θ, β values even in the case of feature correlation. The robot either infers a good θ to perform its original task through the means of another feature, or it has low confidence in understanding the input.

When we look at the posterior distribution that results from all 12 demonstrations, the bottom-right part of the figure shows that, due to the correlation in the blue cluster,

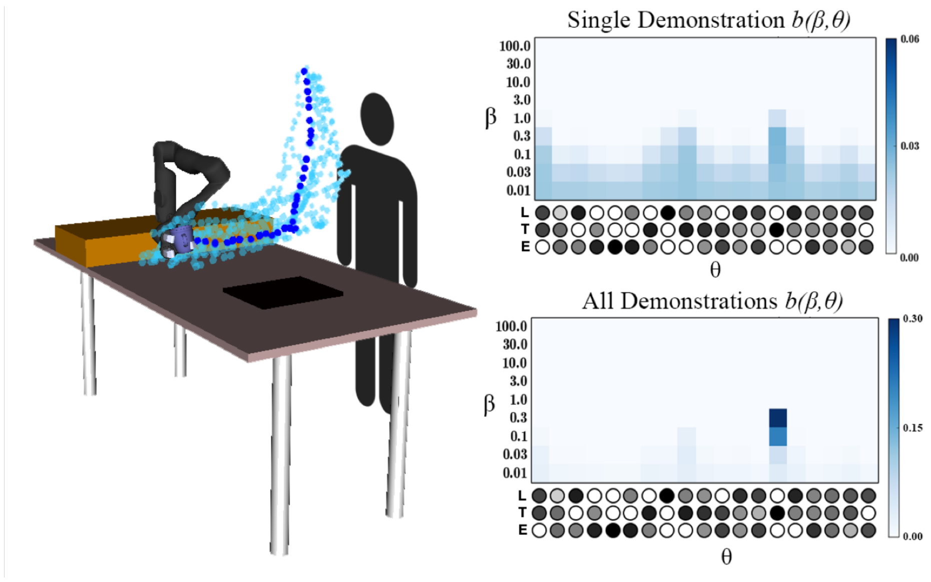


Figure 7.10: (Left) Human demonstrations keeping the cup in the end-effector close to the table. (Right) Because it is difficult for the person to give a good demonstration, the top posterior does not have a clearly defined peak for one particular hypothesis, although several θs are favored. In the bottom distribution, we notice that when presented with all 12 demonstrations, the robot can more clearly infer the correct hypothesis for the distance to the table, $\theta = [0, 1, 0]$.

there is increased probability on the θ that considers fully the distance from the laptop. However, due to the ambiguity of the orange cluster, the situational confidence is not as high as it would be in a well-explained case (see Fig. 7.7).

7.5.1.4 Feature Engineering. Many of the reward function features we considered so far have been intuitive to provide demonstrations for. However, some reward functions may be particularly challenging or unintuitive for human users. Two extreme examples of this could be features learned using complex function approximators or unintuitive features like minimizing the total energy of a system.

In our scenario, the feature users have a difficult time providing good demonstrations for is the distance between the robot's hand and the table along the trajectory. Since the feature was designed as the sum of distances to table for all waypoints in the trajectory, the optimal demonstration immediately moves the end-effector to the table and then keeps it right above the tabletop for the rest of the path, as seen in Fig. 7.3a. This limitation does not support H1.

However, this mathematical optimum does not necessarily align with how human users interpret the best behavior for this task. In our experiments, most users gradually bring the robot's hand closer to the table, rather than pushing it down immediately, for a more smooth and natural motion (see left in Fig. 7.10). These demonstrations thus appear noisy and sub-optimal with respect to the robot's model and make it difficult to infer the true θ from a single demonstration.

This phenomenon is reflected more clearly when we look at the top-right belief distribution in Fig. 7.10. Although the distribution for the highlighted blue demonstration has some peaks around hypotheses that strongly favor the feature responsible for distance to the table, it is not nearly as clearly defined as it should be for a well-explained demonstration (see Fig. 7.7).

However, when the robot gathers evidence from multiple demonstrations, the algorithm does manage to figure out that this is the feature that people were optimizing for. The bottom right plot in Fig. 7.10 illustrates that, once again, having more input samples eventually leads our algorithm to converge to a strong probability for the right θ with a reasonably high β. Although our method cannot back H1 when inferring the objective from a single demonstration, more data leads our algorithm to correctly support H1.

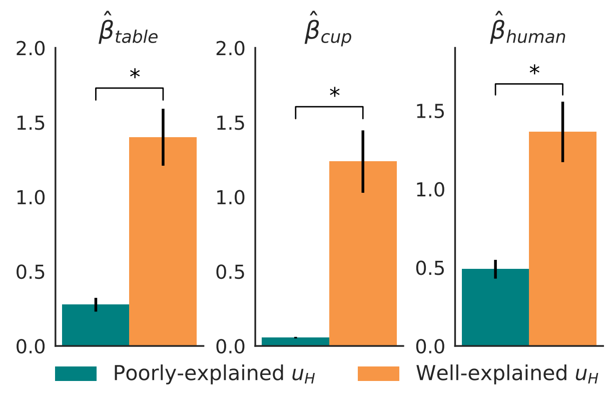
Summary: The four situations presented above illustrate that our two original hypotheses H1 and H2 are supported most of the time (Sec. 7.5.1.1, Sec. 7.5.1.2), with some exceptions (Sec. 7.5.1.3, Sec. 7.5.1.4). We saw that when the person has a difficult time giving a good demonstration (Sec. 7.5.1.4), our method cannot support H1 unless provided with multiple demonstrations, to disambiguate the inherent noise in the user's suboptimal input. Additionally, when the person provides what should be a poorly-explained demonstration (Sec. 7.5.1.3), feature correlation might lead the inference to falsely detect θs corresponding to that input, contradicting H2. However, we observed that when given more demonstrations, our algorithm can attribute low situational confidence β if the uncorrelated input is sufficient. More work is needed in this area.

7.5.2 Corrections

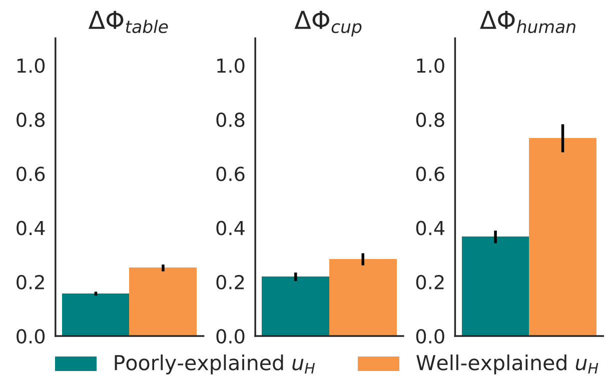
We now turn our attention to case where human input is sparse and in the form of intermediate corrections during the robot's task execution. Here we present an offline case study where we analyze how our estimates of $\hat{\beta}$ enable us to distinguish if the input is well explained or not to the robot's model of the human. For a full exploration of the real-time updates from human corrections, we conduct an online user study which we later describe in Sec. 7.6.

We recruited 12 additional individuals to physically interact with the same robotic manipulator. Each participant was asked to intentionally correct a feature (that the robot may or may not have in its representation): adjusting the distance of the end effector from the table, adjusting the distance from the person, or adjusting the cup's orientation.

During this case study the robot did not attempt to update the feature weights θ and simply tracked a predefined trajectory with an impedance controller [138]. The participants were instructed to intervene only once during the robot's task execution, in order to record a single physical correction. The resulting trajectories and physical



(a) β in well-explained and poorly-explained interactions.



(b) $\Delta\Phi$ in well-explained and poorly-explained interactions.

Figure 7.11: β values are significantly larger for well-explained actions than for poorly-explained ones. Feature updates are non-negligible even during poorly-explained actions, which leads to significant unintended learning for fixed-β methods.

interaction u_H were saved for offline analysis. This setup enabled us to easily analyze the situational confidence of the robot as we changed the robot's representation.

Next, we ran our approximate inference algorithm using the recorded human interaction torques and robot joint angle information. We measured what $\hat{\beta}$ would have been for each interaction if the robot knew about a given subset of the features. By changing the subset of features for the robot, we changed whether any given human interaction was well *explained* to the robot's representation.

We tested two hypotheses:

H1. *Well-explained interactions result in high $\hat{\beta}$, whereas interactions that change a feature the robot* ***does not*** *know about result in low $\hat{\beta}$ for all features the robot* ***does*** *know about.*

H2. *Not reasoning about well-explained interactions and, instead, indiscriminately learning from every update leads to significant unintended learning.*

We ran a repeated-measures ANOVA to test the effect of whether an input is well-explained on our $\hat{\beta}$. We found a significant effect ($F(1, 521) = 9.9093$, $p = 0.0017$): when the person was providing a well-explained correction, $\hat{\beta}$ was higher. This supports H1.

Fig. 7.11a plots $\hat{\beta}$ under the well-explained (orange) and poorly-explained (blue) conditions. Whereas the poorly-explained interactions end up with $\hat{\beta}$s close to 0, well-explained corrections have higher mean and take on a wider range of values, reflecting varying degrees of human performance in correcting something the robot knows about. We fit per-feature chi-squared distributions for $P(\hat{\beta} \mid E)$ which we will use to infer E and, thus, θ online. In addition, Fig. 7.11b illustrates that even for poorly-explained human actions u_H, the resulting feature difference $\Delta\Phi = \Phi(\mathbf{x}_D) - \Phi(\mathbf{x})$ is non-negligible. This supports our second hypothesis, H2, that not reasoning about how well-explained an action is is detrimental to learning performance when the robot receives misaligned updates.

7.6 User Study: Real-time Learning from Corrections

Our case study on corrections suggested that $\hat{\beta}$ can be used as a measure of whether physical interactions are well explained and should be learned from. Next, we conducted an IRB-approved user study to investigate the implications of using these estimates during learning. During each experimental task, the robot began with a number of incorrect weights and participants were asked to physically correct the robot. Locations of the objects and human were kept consistent in our experiments across tasks and users to control for confounds[10]. The planning and inference were done for robot trajectories in 7-dimensional configuration space, accounting for all relevant constraints including joint limits and self-collisions, as well as collisions between obstacles in the workspace and any part of the robot's body.[11]

7.6.1 Experiment Design

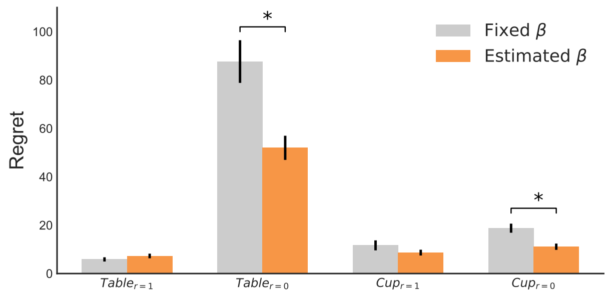
Independent Variables. We used a 2 by 2 factoral design. We manipulated the corrections learning strategy with two levels (fixed-β and estimated-β learning), and also whether the human corrected for features inside (well-explained) or outside (poorly-explained) the robot's representation. In the fixed learning strategy, the robot updated its feature weights from a given interaction via (7.31) with a fixed β value. In the estimated-β learning strategy, the robot updates its feature weights via (7.30). The offline experiments above provided us an estimate for $P(E \mid \hat{\beta})$ that we used in the gradient update.

Dependent Measures - Objective. To analyze the objective performance of the two learning strategies, we focused on comparing two main measurements: the length of the $\hat{\theta}$ path through weight space as a measurement of the learning process, and the regret in feature space measured by $|\Phi(\mathbf{x}_{\theta^*}) - \Phi(\mathbf{x}_{actual})|$. Longer $\hat{\theta}$ paths should indicate a learning process that oscillates, whereas shorter paths suggest smoother learning curves. On the other hand, high regret implies that the learning method did not converge to a good objective θ, whereas low regret indicates better learning.

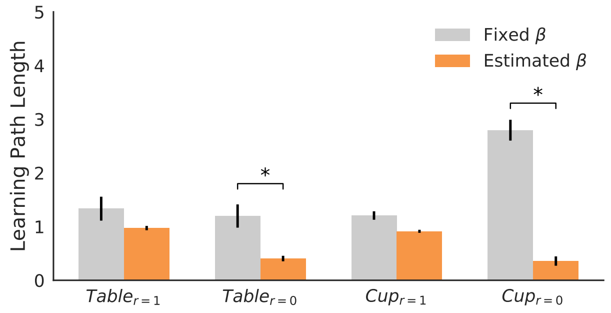
Dependent Measures - Subjective. For each condition, we administered a 7-point Likert scale survey about the participant's interaction experience (Table 7.1). We separate the survey into 3 scales: task completion, task understanding, and unintended learning.

Hypotheses.
H1. *On tasks where humans try to correct* ***inside*** *the robot's representation (well-explained corrections), inferring situational confidence is not inferior to assuming high situational confidence.*
H2. *On tasks where humans try to correct* ***outside*** *the robot's representation (poorly-explained corrections), inferring situational confidence reduces unintended learning.*

[10]We assume full observability of the objects and the human, as the focus of this paper is not sensing.
[11]For video footage of the experiment, see: https://youtu.be/stnFye8HdcU



(a) Regret averaged across subjects.



(b) $\hat{\theta}$ learning path length averaged across subjects.

Figure 7.12: Comparison of regret and length of $\hat{\theta}$ learning path through weight space over time.

H3. *On tasks where they tried to correct **inside** the robot's representation, participants felt like the two methods performed the same.*
H4. *On tasks where they tried to correct **outside** the robot's representation, participants felt like our estimated-β method reduced unintended learning.*

Tasks. We designed 4 experimental household motion planning tasks for the robot to perform in a shared workspace. Similarly to the case studies, for each experimental task the robot carried a cup from start to end with an initially incorrect reward. Participants were instructed to physically intervene to correct the robot's behavior during the task.

In Tasks 1 and 2, the robot's default trajectory took a cup from the participant and put it down on the table, but carried the cup too high above the table. In Tasks 3 and 4, the robot also took a cup from the human and placed it on the table, but this time it initially grasped the cup at the wrong angle, requiring human assistance to correct end-effector orientation to an upright position. For Tasks 1 and 3, the robot knew about the feature the human was asked to correct for ($E = 1$) and participants were told that the robot should be compliant. For Tasks 2 and 4, the correction was poorly explained ($E = 0$) and participants were instructed to correct any additional unwanted changes in the trajectory.

Participants. We used a within-subjects design and randomized the order of the learning methods during experiments. We recruited 12 participants (6 females, 6 males, aged 18-30) from the campus community, 10 of which had technical background. None of the participants had experience interacting with the robot used in our experiments.

Procedure. Every participant was assigned a random ordering of the two methods, and performed each task without knowing how the underlying methods work. One challenge in running our experiment was that different participants may have different internal preferences for how a task should be performed. In order to have a consistent notion of ground-truth preferences, we fixed the true objective (e.g. how far the cup should be from the table) for each task. At the beginning of each task, the participant was first shown the incorrect default trajectory that they must correct, followed by the ground-truth desired trajectory they should teach the robot. This allows us to focus on how well each algorithm

	Questions	Cronbach's α	F-Ratio	p-value
task	The robot accomplished the task in the way I wanted. The robot was NOT able to complete the task correctly.	0.94	0.88	0.348
understand	I felt the robot understood how I wanted the task done. I felt the robot did NOT know how I wanted the task done.	0.95	0.55	0.46
unintend	I had to undo corrections that I gave the robot. The robot wrongly updated its understanding about aspects of the task I did not want to change. After I adjusted the robot, it continued to do the other parts of the task correctly. After I adjusted the robot, it incorrectly updated parts of the task that were already correct.	0.91	9.15	**0.0046**

Table 7.1: Results of ANOVA on subjective metrics collected from a 7-point Likert-scale survey.

infers objectives from human input, versus trying to additionally estimate the unique ground-truth human objective of each participant. Then the participant performed a familiarization round, followed by two recorded experimental rounds. After answering the survey, the participant repeated the procedure for the other method.

7.6.2 Analysis

Objective. We ran a repeated-measures factorial ANOVA with learning strategy and input quality (well or poorly explained) as factors. We found a significant main effect for the method ($F(1,187) = 7.8, p = 0.0058$), and a significant interaction effect ($F(1,187) = 6.77, p = 0.0101$). We ran a post-hoc analysis with Tukey HSD corrections for multiple comparisons to analyze this effect, and found that it supported our hypotheses. On tasks where corrections were poorly explained, the estimated-β method had significantly lower regret ($p = 0.001$); on tasks where corrections were well explained, there was no significant difference ($p = 0.9991$). Fig. 7.12a plots the regret per task, and indeed the estimated-β method was not inferior on tasks 1 and 3, and significantly better on tasks 2 and 4.

For the length of the $\hat{\theta}$ path metric, the factorial ANOVA analysis found a significant main effect for the method ($F(1,187) = 76.43, p < 0.0001$), and a significant interaction effect ($F(1,187) = 33.3, p < 0.0001$). A similar post-hoc analysis with Tukey HSD correction for multiple comparisons also supports our hypotheses. On tasks where corrections were poorly explained, our method had lower average weight paths over time ($p = 0.0025$); on tasks where correction were well explained, however, there was no significant difference ($p = 0.1584$). The same results are supported by Fig. 7.12b, which plots the average length of $\hat{\theta}$ through weight space per task, and indeed our method was not significantly inferior for tasks 1 and 3, and significantly better on tasks 2 and 4.

Subjective. We ran a repeated measures ANOVA on the results of our participant survey. We find that our method is not significantly different from the baseline in terms of task completion ($F(1,7) = 0.88, p = 0.348$) and task understanding ($F(1,7) = 0.55, p = 0.46$), which supports H3. Participants also significantly preferred the estimated-β method in terms of reducing unintended learning ($F(1,7) = 9.15, p = 0.0046$), which supports H4.

7.7 Discussion

Human guidance is becoming increasingly important as autonomous systems enter the real world. One common way for robots to interpret human input is treating it as evidence about hypotheses in the robot's reward space. Since accounting for all possible hypotheses and situations ahead of time is challenging if not infeasible, in this paper we claim that robots should explicitly reason about how well their given representation can explain the human inputs.

We introduced the notion of *situational confidence* β as a natural way to measure how much the robot should trust its inputs and learn from them. We presented a general framework for estimating β in conjunction with any task objectives for scenarios where the human and the robot are operating the same dynamical system. We instantiated it for learning from human demonstrations, as well as for learning from corrections, by deriving a close to real-time approximate algorithm. In both settings, we exemplified – via human experiments with a 7-DoF robotic manipulator and a user study – that reasoning about situational confidence does, in fact, assist the robot in better understanding when it cannot explain human input.

There are several important limitations in our work. Perhaps the biggest limitation of all, which we alluded to in the introduction, is that the representation can be misaligned but the robot can nonetheless explain the input relatively well, thus confusing misalignment for slight noise. This is especially true in more expressive representations, where there might always be some hypothesis that explains the input. This is unfortunately a fundamental problem with detecting misalignment in expressive representations: a single demonstration or a single data point will not be enough. Much like learning cost functions when using such spaces requires much more and diverse data than when using a less expressive space, with detecting misalignment too it will be the case that the robot will require a rich and diverse set of data points. The more data the robot has access to, and the more diversely it is distributed, the less of a chance there is that one wrong hypothesis can explain all the data. Furthermore, our approach cannot disambiguate between misalignment of the representation and misalignment of the human observation model, i.e. the Boltzmann model.

Algorithmically, while for corrections we derived a way to handle continuous reward spaces that scales linearly with the dimensionality of the space, for demonstrations we relied on simply discretizing the space. This was sufficient for showcasing the benefit of estimating situational confidence, since for demonstrations this is done offline. However, to scale the method to complex spaces, we need to combine it with state-of-the-art IRL approaches that rely on Metropolis Hastings sampling, or simply estimate the MLE. Lastly, experimentally for both demonstrations and corrections we are limited to a simple motion planning task with a cost function that depends on only a few features.

In subsequent work, we hope to address some of these limitations. We are also interested in an extension to sequential time-dependent inputs, where the person could change their mind about what objective is important to them. Additionally, we want to explore ways of handling misalignment other than reducing learning, such as switching to a more expressive representation (but demanding more data and computation) whenever the situational confidence is very low for all θs. Finally, we are excited to showcase our work on other coupled dynamical systems, such as autonomous vehicles.

7.8 Additional Details and Practical Considerations

7.8.1 Demonstrations

Discretizing Θ and $\mathcal{B}$ in (7.13). For the Θ discretization, we chose vectors in the unit sphere, as discussed in Sec. 7.2.2. For practical purposes, we restricted the θ components to be positive due to our task features and the capabilities of our trajectory optimizer; in general, learning from demonstrations should be restricted to norm 1, not necessarily to the positive quadrant. In both our examples in Sec. 7.3 and experiments in Sec. 7.5, each θ_i component was allowed to take values 0, 0.5, or 1. Since we used 3 features, θ's dimensionality was 3, leading to a possible set Θ equivalent to the 3-fold Cartesian product of the values above. After normalizing to norm 1, we were left with 19 unique θ vectors in Θ, weighing the three features in different proportions, as shown in Figures 7.3, 7.7, 7.8, 7.9, and 7.10. Our discretization scheme ensured an approximately uniform sampling on the positive quadrant of the unit sphere.

To discretize situational confidence, we found it sufficient to cover the log-scale space, similarly to [100, 98]: $\beta \in \{0.01, 0.03, 0.1, 0.3, 1.0, 3.0, 10.0, 30.0, 100.0\}$. For different tasks, a similar discretization should suffice because what matters is β's relative magnitude for identifying misalignment, not its absolute one. We suggest calibrating the threshold ϵ in (7.6) using a few simulated trajectories like the ones in Fig. 7.3.

7.8.2 Corrections

Planning and Replanning. We use TrajOpt [255] to plan and replan robot trajectories. We set up the trajectory optimization problem to plan a path that minimizes a cost given by the negative of the reward function (7.15). Given different features Φ and weights θ on these features, different optimal paths may be found. Additionally, we constrain the optimization to plan a path between a pre-specified start and goal locations, while avoiding collisions with the objects in the environment (table, laptop, or human). The total time of the trajectory is fixed, but the actual length can differ. That means that the robot moves faster for longer trajectories, and slower for shorter ones.

The robot plans an initial path from start to goal using the initial weights θ. When a human pushes, the robot measures the instantaneous deviation, which deforms the trajectory via the impedance controller. Without learning, the robot would resume tracking its original trajectory. However, we use the human input to update θ according to (7.30), which the robot's planner uses to compute a new trajectory that the robot can follow instead. In a perfect world, this entire process would happen at 60Hz. In practice, however, the trajectory optimizer's computation lasts longer. As such, once a push is registered, the robot starts listening for following torque signals only after the update is complete.

Imagine this process in the context of a typical user experience. Once the person begins pushing, the robot instantly starts updating θ and optimizing the new induced path. While the person is applying their correction, the planner eventually finishes its computation and passes the updated trajectory to the robot controller. The user can immediately feel that the robot changed course and stops intervening.

Solving (7.21). We used SLSQP, an off-the-shelf sequential quadratic programming package [163], to solve (7.21). In practice, the method can fail to return a good result if the initialization is bad. We found that if we initialize the minimization with a guess that does not satisfy the constraint (e.g. 0), it returns a reasonable estimate of the true u_H^*.

Sensitivity Analysis. Both (7.24) and (7.30) rely heavily on λ and ν.

Setting λ affects the magnitude of the resulting estimated situational confidence $\hat{\beta}$ in (7.24). This magnitude plays an important role when later estimating θ via (7.30) because it affects $P(E \mid \hat{\beta})$. However, note that to compute this probability we use $P(\hat{\beta} \mid E)$, which is an entirely data-driven empirical distribution, where the observed $\hat{\beta}$ is also computed via (7.24). As such, we are not relying on absolute magnitudes of the estimated situational confidence but on relative ones. Therefore, the choice of the hyperparameter λ does not affect our method's estimates as long as they are computed with the same hyperparameter that is used for learning $P(\hat{\beta} \mid E)$.

In the case of precision ν in (7.25), how spread out the Gaussian noise centered around $\Phi(\mathbf{x}_R)$ is affects the denominator in (7.30). When $\nu \to 0$, the $\Gamma(\Phi_D, E = 0)$ term in the denominator goes to 0, which means that (7.30) reduces to (7.31): our method always learns and never identifies misalignment. On the other hand, when $\nu \to \infty$, we can use the L'Hospital rule to see that $\Gamma(\Phi_D, E = 0) \to 0$ as well, as long as $||\Phi_D - \Phi(\mathbf{x}_R)||^2 \neq 0$, which is true unless there is no correction to deform $\mathbf{x}_R$, in which case we do not need to update θ at all. Therefore, it is important that ν is set not too high and not too low in order for our method to work properly.

The best practice for setting ν also involves using the offline data calibration from Sec. 7.5.2: after computing the empirical $P(\hat{\beta} \mid E)$ distribution, when $E = 0$ the updated θ should not change much, whereas when $E = 1$ the θ parameter should change appropriately. Without the offline data calibration in Sec. 7.5.2, both λ and ν affect the θ and β estimation, and can have profound effects on the efficacy of our method. Unfortunately, we cannot do this calibration automatically yet, which is a limitation of our work.

Trajectory Deformation Parameter Choice. When deforming the robot's trajectory given a human interaction, there are many choices of the deformation matrix A and the deformation magnitude parameter μ. A can be an explicit design choice (for example, constructing A from a finite differencing matrix [25]), can be solved for via an optimization problem which penalizes the undeformed trajectory's energy, the work done by the trajectory deformation to the human, and variation's total jerk as in [191], or can even be learned from human data [147]. The magnitude of the deformation μ can also be tuned for best performance, for example to be robust to the rate at which deformations occur (see [190] for more details).

7.8.3 Laplace Approximation in Equation (7.19)

Let the cost in the denominator in (7.19) be denoted by:

$$C_{\Phi_D}(\bar{u}) = \lambda\|\bar{u}\|^2 + \kappa\|\Phi(\bar{\mathbf{x}}_D) - \Phi_D\|^2, \tag{7.32}$$

for an observed Φ_D.

First, our cost function can be approximated to quadratic order by computing a second order Taylor series approximation about the optimal human action u_H^* (obtained via the constrained optimization in (7.21)):

$$C_{\Phi_D}(\bar{u}) \approx C_{\Phi_D}(u_H^*) + \nabla C_{\Phi_D}(u_H^*)^\top(\bar{u} - u_H^*) + \frac{1}{2}(\bar{u} - u_H^*)^\top \nabla^2 C_{\Phi_D}(u_H^*)(\bar{u} - u_H^*) \ . \tag{7.33}$$

Since $\nabla C_{\Phi_D}(\bar{u})$ has a global minimum at u_H^* then $\nabla C_{\Phi_D}(u_H^*) = 0$ and the denominator of (7.19) can be rewritten as:

$$\int_{\mathcal{U}} e^{-\beta C_{\Phi_D}(\bar{u})} d\bar{u} \approx e^{-\beta C_{\Phi_D}(u_H^*)} \int_{\mathcal{U}} e^{-\frac{1}{2}(\bar{u}-u_H^*)\beta\nabla^2 C_{\Phi_D}(u_H^*)(\bar{u}-u_H^*)} d\bar{u} \ . \tag{7.34}$$

Since $\beta\nabla^2 C_{\Phi_D}(u_H^*) > 0$ for $u_H^* \neq 0$, the integral is in Gaussian form, which admits a closed form solution:

$$\int_{\mathcal{U}} e^{-\beta C_{\Phi_D}(\bar{u}_H)} d\bar{u}_H \approx e^{-\beta C_{\Phi_D}(u_H^*)} \sqrt{\frac{2\pi^k}{\beta^k |H_{u_H^*}|}} \ ,$$

where $H_{u_H^*} = \nabla^2 C_{\Phi_D}(u_H^*)$ denotes the Hessian of C_{Φ_D} at u_H^*. Replacing $C_{\Phi_D}(\bar{u}_H)$ with the expanded cost function, we arrive at the final approximation of the observation model:

$$P(u_H^t \mid x^0, \mathbf{u}_R; \Phi_D, \beta) \approx \frac{e^{-\beta\lambda(\|u_H^t\|^2)}}{e^{-\beta(\lambda\|u_H^*\|^2 + \kappa\|\Phi(\mathbf{x}_D^*) - \Phi_D\|^2)}} \sqrt{\frac{\beta^k |H_{u_H^*}|}{2\pi^k}} \ .$$

Chapter 8

Misalignment Detection in Teleoperation

This chapter is based on the paper "Situational Confidence Assistance for Lifelong Shared Autonomy" [313], written in collaboration with Matthew Zurek, Daniel Brown, and Anca Dragan.

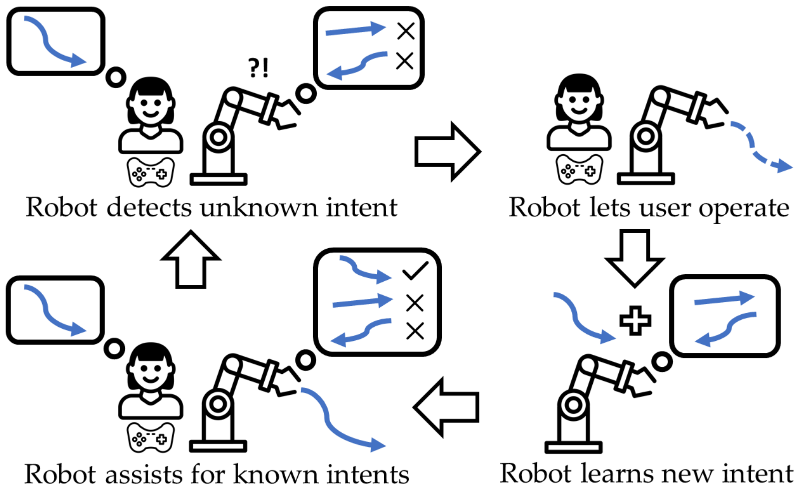


Figure 8.1: We propose an approach for lifelong shared autonomy that enables a robot to detect when its set of known human intents is insufficient to explain the current human behavior. Rather than trying to assist for the wrong intent, the robot learns from novel teleoperations to learn a model of the new intent, allowing for lifelong confidence-based assistance.

In shared autonomy [9, 89, 240, 192, 110, 5, 189, 181, 92, 52, 75], robots assist human operators to perform their objectives more effectively. Here, rather than directly executing the human's control input, a typical framework has the robot estimate the human's intent and execute controls that help achieve it [89, 145, 210, 225, 240].

These methods succeed when the robot knows the set of possible human intents a priori, e.g. the objects the human might want to reach, or the buttons they might want to push [89, 145]. But realistically, users of these systems will inevitably want to perform tasks outside the repertoire of known intents – they might want to reach for a goal unknown to the robot, or perform a new task like pouring a cup of water into a sink. This presents a three-fold challenge for shared autonomy. First, the robot will be unable to recognize and help with something unknown. Second, and perhaps more importantly, it will attempt to assist with whatever wrong intent it infers, interfering with what the user is trying to do and hindering their performance. This happens when the robot plans in expectation [145], and, as our experiments will demonstrate, it happens even when the robot arbitrates the amount of assistance based on its confidence in the most likely goal [89]. Third, the new task remains just as difficult as the first time even after arbitrarily many attempts.

Our key idea is that the robot should detect that the user is trying something new and give them control. This then presents an opportunity for the robot to observe the new executed trajectory, learn the underlying intent that explains it, and add it to its repertoire so that it can infer and assist for this intent in the future.

To achieve this, we need two ingredients: 1) a way for the robot to detect its repertoire of intents is insufficient, and 2) a representation of intents that enables learning new tasks throughout its lifetime, adding them to its repertoire, and performing inference over them in a unified way with the initial known intents. For the latter, we use reward functions to unify goals and general skills like pouring into the same representation. This then enables the former: when the human acts too suboptimally for any of the known reward functions, it suggests the robot lacks the correct set of rewards.

Our approach takes inspiration from Chapter 7 on representation misalignment where the robot recognizes when its reward function features are insufficient to explain human demonstrations and corrections [38], and updates the reward in proportion to the *situational confidence* in these features' ability to explain input. We extend detecting representation misalignment to the context of shared autonomy, in which there are multiple intents, represented as reward functions, and the robot seeks to recognize whether any of the known intents explain the human input sufficiently. The robot can then arbitrate its assistance based on its confidence in the most likely intent being what the human wanted.

Our approach, which we call Confidence-Aware Shared Autonomy (CASA), allows the robot to ascertain whether the human inputs are associated with a known or new task. By arbitrating the user's input based on the confidence in the most likely intent, CASA follows a standard policy blending assistance approach if the task is known, and otherwise gives the user full control. Additionally, CASA allows the user to provide a few demonstrations of the new intent, which the robot uses to learn a reward function via Inverse Reinforcement Learning (IRL) [96] and add it to its set of intents. This enables lifelong shared autonomy, where the robot helps when it is confident in what the user wants and learns new intents when it detects that the human is doing something novel, so that it can assist with that intent in the future.

We test our approach in a expert case study and a user study with a simulated 7DoF JACO assistive robot arm. Our results suggest that CASA significantly outperforms prior approaches when assisting for unknown intents, maintains high performance in the case of known ones, and successfully learns new intents for better lifelong shared autonomy.

8.1 Approach: Confidence-Aware Shared Autonomy

We consider a human teleoperating a dexterous robot arm to perform everyday manipulation tasks. The robot's goal is to assist the person in accomplishing their desired skill by augmenting or changing their input. While the robot possesses a predefined set of possible intents, the human's desired motion might not be captured by any of them. We propose that since the robot might not understand the person's intentions, it should reason about how confident it is in its predictions to avoid assisting for the wrong intent.

8.1.1 Preliminaries

Formally, let $s \in \mathcal{S}$ be the continuous robot state (e.g. joint angles), and $a \in \mathcal{A}$ the continuous robot action (e.g. joint velocity). The user controls their desired robot configuration by providing continuous inputs $a_H \in \mathcal{A}_\mathcal{H}$ via an interface (e.g. GUI, joystick, keyboard commands, etc). These inputs are mapped to robot actions through a *direct teleoperation* function $\mathcal{T} : \mathcal{A}_\mathcal{H} \rightarrow \mathcal{A}$. Define a person's trajectory up until time t as the sequence $\xi_{0\rightarrow t} = (s^0, a_H^0, \dots, s^t, a_H^t)$.

The robot is equipped with a set of known intents Θ, one of which may represent the user's desired motion. Each intent is parameterized by a reward function R_θ, which may be hand-engineered or learned from demonstrations via IRL [311, 214]. For example, if the intent represents moving to a goal g, the reward function can be given by the distance to the goal: $R_g(\xi) = -\sum_{x\in\xi} \|x - g\|$. If the intent is pouring a cup, the reward can be a neural network with parameters ψ, R_ψ. Our shared autonomy system does not know the intent a priori, but infers it from the human's inputs. Given the user's trajectory so far, $\xi_{0\rightarrow t}$, a common strategy is to predict the user's intent $\theta \in \Theta$, compute the optimal action for moving accordingly, then augment the user's original input with it [89].

However, what if none of the intents match the human's input, i.e., the person is doing something the robot does not know about? We introduce a shared autonomy formalism where the robot reasons about its confidence in its current set of intents' ability to explain the person's input, and uses that confidence for assistance. This confidence serves a dual purpose, as the robot can also use it to ask the human to demonstrate the missing intent.

8.1.2 Intent Inference

To assist the person, the robot has to first predict which of its known tasks the person is trying to carry out, if any. To do that, the robot needs a model of how people teleoperate it to achieve a desired motion. We assume the Boltzmann noisily-rational decision model [27]:

$$P(\xi \mid \theta, \beta) = \frac{e^{\beta R_\theta(\xi)}}{\int_{\bar{\xi}} e^{\beta R_\theta(\bar{\xi})} d\bar{\xi}} \ , \tag{8.1}$$

where the person chooses the trajectory ξ proportionally to its exponentiated reward R_θ. The parameter $\beta \in [0, \infty)$ controls how much the robot expects to observe human input consistent with the intent θ. Typically, β is fixed, recovering the Maximum Entropy IRL observation model [311], which is what most inference-based shared autonomy methods use [89, 145]. Inspired by work on confidence-aware human-robot interaction [100, 98, 38], we instead reinterpret β as the robot's *situational confidence* in its ability to explain human data, given the known intents Θ, and we show how the robot can estimate it in Sec. 8.1.3.

Given (8.1), if the reward R_θ of intent θ is additive along the trajectory ξ, we have that:

$$P(\xi_{0\to t} \mid \theta, \beta) = e^{\beta R_\theta(\xi_{0\to t})} \frac{\int_{\bar{\xi}_{t\to T}} e^{\beta R_\theta(\bar{\xi}_{t\to T})}}{\int_{\bar{\xi}_{0\to T}} e^{\beta R_\theta(\bar{\xi}_{0\to T})}} \;, \tag{8.2}$$

where T is the episode length. In high-dimensional manipulation spaces, evaluating these integrals is intractable. We follow [89] and approximate them via Laplace's method:

$$P(\xi_{0\to t} \mid \theta, \beta) \approx e^{\beta\left(R_\theta(\xi_{0\to t}) + R_\theta(\xi^*_{t\to T}) - R_\theta(\xi^*_{0\to T})\right)} \sqrt{\left(\frac{\beta}{2\pi}\right)^{tk} \frac{|\nabla^2 R_\theta(\xi^*_{0\to T})|}{|\nabla^2 R_\theta(\xi^*_{t\to T})|}} \;, \tag{8.3}$$

where k is the action dimensionality, and the trajectories $\xi^*_{0\to T}$ and $\xi^*_{t\to T}$ are optimal with respect to R_θ and can be computed with any off-the-shelf trajectory optimizer[1].

Now, given a tractable way to compute the likelihood of the human input, the robot can obtain a posterior over intents:

$$P(\theta \mid \xi_{0\to t}, \beta) = \frac{P(\xi_{0\to t} \mid \theta, \beta)}{\sum_{\theta' \in \Theta} P(\xi_{0\to t} | \theta', \beta)}, \tag{8.4}$$

assuming $P(\theta \mid \beta) = P(\theta)$ and a uniform prior over intents.

Prior inference-based shared autonomy work [89, 145] typically assumes $\beta = 1$. We show that the robot should not be restricted by such an assumption and it, in fact, benefits from estimating $\hat{\beta}$ and reinterpreting it as a confidence.

8.1.3 Confidence Estimation

In the Boltzmann model in (8.1), we see that β determines the variance of the distribution over human trajectories. When β is high, the distribution is peaked around those trajectories ξ with the lowest reward R_θ; in contrast, a low β makes all trajectories equally likely. We can, thus, reinterpret β to take a useful meaning in shared autonomy: given an intent, β controls how well that intent's reward explains the user's input. A high β for an intent θ indicates that the intent's reward explains the input well and is a good candidate

[1]We use TrajOpt [255], based on sequential quadratic programming.

for assistance. A low β on all intents suggests that the robot's intent set is insufficient for explaining the person's trajectory.

We estimate β and use it for assistance. We use the model (8.3) to write the β posterior

$$P(\beta \mid \xi_{0\to t}, \theta) = \frac{P(\xi_{0\to t} \mid \theta, \beta)P(\beta)}{\int_{\bar{\beta}} P(\xi_{0\to t}|\theta, \bar{\beta})P(\beta)d\bar{\beta}}. \tag{8.5}$$

If we assume a uniform prior $P(\beta)$, we may compute an estimate of the confidence parameter β per intent θ via a maximum likelihood estimate:

$$\hat{\beta}_\theta = \arg\max_{\bar{\beta}} e^{\bar{\beta}\left(R_\theta(\xi_{0\to t})+R_\theta(\xi^*_{t\to T})-R_\theta(\xi^*_{0\to T})\right)} \left(\frac{\bar{\beta}}{2\pi}\right)^{\frac{tk}{2}} \quad , \tag{8.6}$$

where we drop the Hessians since they don't depend on β. Setting the derivative of the objective in (8.6) to zero and solving for β yields the following estimate:

$$\hat{\beta}_\theta^{MLE} = \frac{tk}{2(R_\theta(\xi^*_{0\to T}) - R_\theta(\xi_{0\to t}) - R_\theta(\xi^*_{t\to T}))} \quad . \tag{8.7}$$

Alternatively, we chose to add an exponential prior with parameter λ, $Exp(\lambda)$ to obtain

$$\hat{\beta}_\theta^{MAP} = \frac{tk}{2(\lambda + R_\theta(\xi^*_{0\to T}) - R_\theta(\xi_{0\to t}) - R_\theta(\xi^*_{t\to T}))} \quad . \tag{8.8}$$

The denominators in equations 8.7 and 8.8 can be interpreted as the "suboptimality" of the observed partial trajectory $\xi_{0\to t}$ compared to the reward of the optimal trajectory for the particular θ, $R_\theta(\xi^*_{0\to T})$. Note that $\hat{\beta}_\theta$ is inversely proportional to the suboptimality divided by the number of time steps t that have passed. Intuitively, the user has more chances to be a suboptimal teleoperator as time goes on, so dividing for t corrects for the natural increase in suboptimality over time.

If this normalized suboptimality is low for an intent θ, then the person is close to a good trajectory for that intent and $\hat{\beta}_\theta$ will be high. Thus, a high $\hat{\beta}_\theta$ means that the person's input is well-explained by that intent. On the other hand, high suboptimality per time means the person is far from good trajectories, so θ's reward model R_θ does not explain the person's trajectory and $\hat{\beta}_\theta$ will be low.

8.1.4 Confidence-based Arbitration

Armed with a confidence estimate $\hat{\beta}_\theta$ for every $\theta \in \Theta$, the robot can predict the most likely one $\theta^* = \arg\max_{\theta\in\Theta} P(\theta \mid \xi_{0\to t}, \hat{\beta}_\theta)$ using (8.4). From here, one natural style of assistance is "policy blending" [89]. First the robot computes an optimal trajectory under the most likely intent, $\xi^* = \arg\max_\xi \sum_{s\in\xi} R^*_\theta(s)$, the first action of which is a^*. Then the robot combines a^* and $\mathcal{T}(a_H^t)$ using a blending parameter $\alpha \in [0, 1]$, resulting in the robot action $a^t = \alpha\mathcal{T}(a_H^t) + (1 - \alpha)a^*$. We also refer to α as the human's control authority.

Prior work proposes different ways to arbitrate between the robot and human actions by choosing α proportional to the robot's distance to the goal or to the probability of the most likely goal [89]. However, when using $P(\theta^* \mid \xi)$, θ^* might look much better than the other intents, resulting in the robot wrongly assisting for θ^*. Distance-based arbitration ignores the full history of the user's input and can only accommodate simple intents.

Instead, we propose that the robot should use its confidence in the most likely intent, $\hat{\beta}_{\theta^*}$, estimated according to Sec. 8.1.3, to control the strength of its arbitration:

$$a^t = \min(1, 1/\hat{\beta}_{\theta^*})\mathcal{T}(a_H^t) + (1 - \min(1, 1/\hat{\beta}_{\theta^*}))a^* \tag{8.9}$$

When $\hat{\beta}_{\theta^*}$ is high, i.e. the robot is confident that the predicted intent θ^* can explain the person's input, α is low, giving the robot more influence through its action a^*. When $\hat{\beta}_{\theta^*}$ is low, i.e. not even the most likely intent explains the person's input, α increases, giving the person's action a_H^t more authority.

8.1.5 Using Confidence for Lifelong Learning

Estimating the confidence $\hat{\beta}_\theta$ also lets the robot detect *misalignment* in Θ: if all estimated $\hat{\beta}_\theta$ for $\theta \in \Theta$ are below a threshold ϵ, the robot is missing the person's intent. Once the robot has identified that its intent set is misaligned, it should ask the person to teach it. We represent the missing intent θ_ϕ as a neural network reward parameterized by ϕ and learn it via deep maximum entropy IRL [96]. The gradient of the IRL objective with respect to the reward parameters ϕ can be estimated by: $\nabla_\phi \mathcal{L} \approx \frac{1}{|\mathcal{D}^*|}\sum_{\tau \in \mathcal{D}^*} \nabla_\phi R_\phi(\tau) - \frac{1}{|\mathcal{D}^\phi|}\sum_{\tau \in \mathcal{D}^\phi} \nabla_\phi R_\phi(\tau)$. $\mathcal{D}^*$ are demonstrations of the person executing the missing intent via direct teleoperation, and $\mathcal{D}^\phi$ are trajectories sampled from the R_ϕ induced near the optimal policy.

Once we have a new intent θ_ϕ, the robot updates its intent set $\Theta \leftarrow \Theta \cup \theta_\phi$. The next time the person needs assistance, the robot can perform confidence estimation, goal inference, and arbitration as before, using the new library of intents. While the complexity scales linearly with $|\Theta|$, planning can be parallelized across each intent.

Learned rewards fit naturally into our framework, allowing for a simple way to compare against the known intents. One could also imagine adapting our method to other ways to learn an intent, from imitation learning [136, 241] to dynamic movement primitives [220]. For instance, if we model intents via policies, we can derive a similar confidence metric based on probabilities of human actions under a stochastic policy rather than rewards.

8.2 Experiments: Confidence Estimation Case Study

We now introduce three manipulation tasks and use expert data to analyze confidence estimation and assistance. In Sec. 8.3 we will test CASA's assistive capacity in a user study.

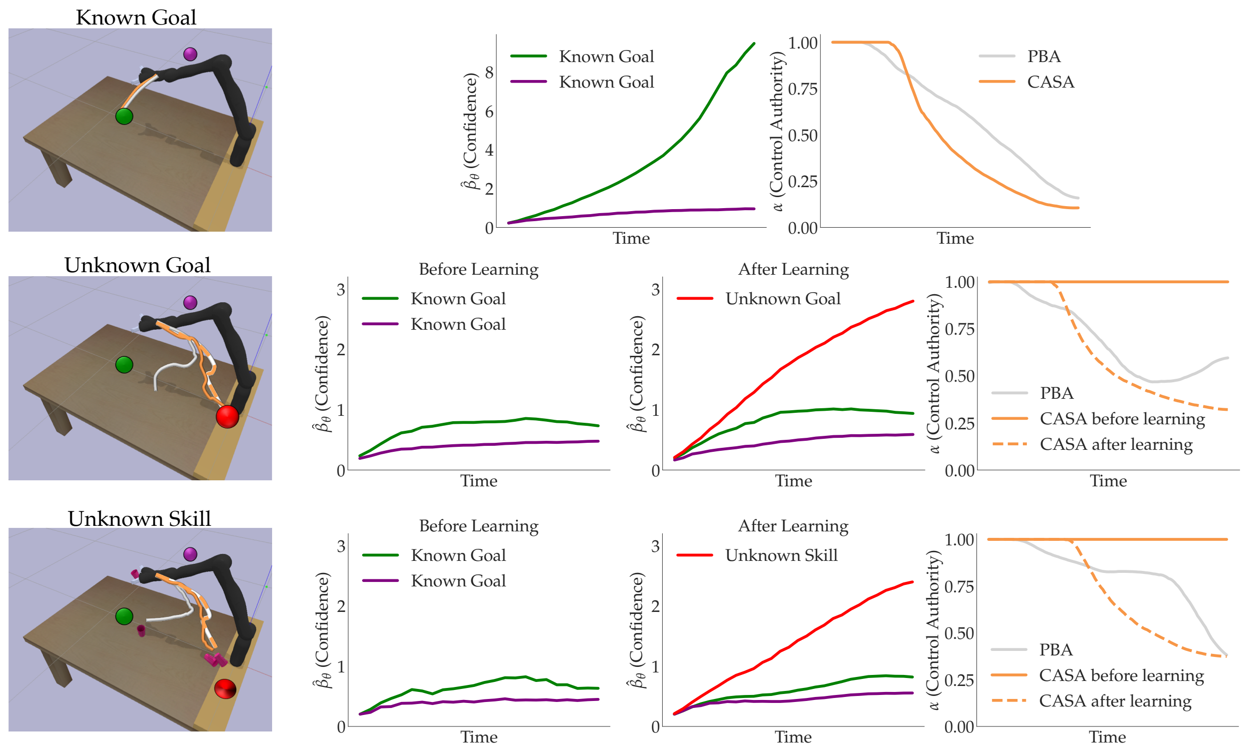


Figure 8.2: Expert case study results. For each task we compute confidence estimates before learning and, for the misaligned tasks (middle, bottom), we recompute the confidence estimates after learning. We plot the strength of assistance before and after learning and compare to a policy blending baseline [89].

8.2.1 Experimental Setting

We conduct our experiments on the simulated 7-DoF JACO arm shown in Fig. 8.2. We use the pybullet interface [74] and teleoperate the robot via keypresses. We map 6 keys to bi-directional xyz movements of the robot's end-effector, and 2 keys for rotating it in both directions. We performed inference and confidence estimation twice per second.

We test CASA on 3 different tasks. In the *Known Goal* task, we control for the well-aligned setting: the robot must assist the user to move to the known green goal location in Fig. 8.2. In the other tasks, we test CASA's efficacy in the case of misalignment, where the user's desired intent is initially missing from the robot's known set Θ. In the second task, *Unknown Goal*, the person teleoperates the robot to the red goal which is unknown to the robot. Finally, in the third and most complicated task, *Unknown Skill*, the person tries to pour the cup contents at an unknown goal location.

For the Unknown Goal and Unknown Skill tasks, we first run CASA before learning the new intent (CASA *before learning*). Detecting low confidence, the robot asks for demonstrations to learn the missing intent via IRL. We then teleoperate with CASA *after learning* to assess the quality of robot assistance after learning the new intent.

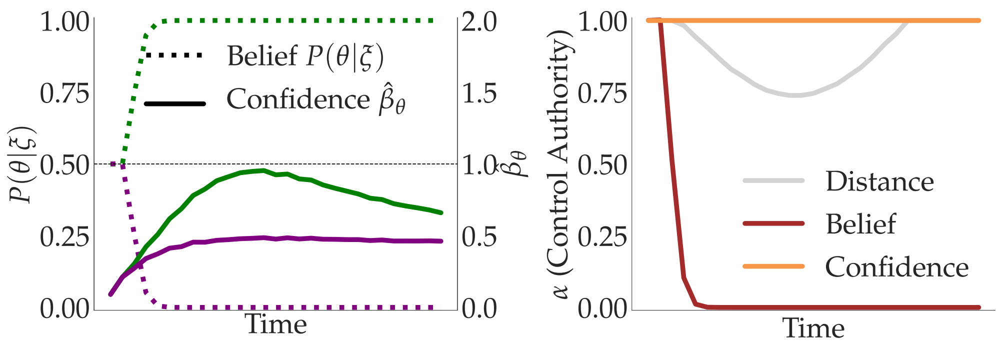


Figure 8.3: Analysis of arbitration methods. After tracking an optimal trajectory for the Unknown Goal task, we show the robot's belief and confidence estimates for each known goal (left), as well as the α values under the distance, belief, and confidence-based arbitration schemes (right).

8.2.2 Arbitration Method Comparison

We compare CASA to a policy blending assistance (PBA) baseline [89] that assumes $\beta = 1$ for all intents. PBA arbitrates with the distance d_{θ^*} to the predicted goal: $\alpha = \min(1, d_{\theta^*}/D)$, with D some threshold past which the robot does not assist. More sophisticated arbitration schemes use $P(\theta^* \mid \xi)$ or the full distribution $P(\theta \mid \xi)$, but they are less robust to misalignment. This is because when the user teleoperates for an unknown intent, $P(\xi \mid \theta)$ will be low for all known $\theta \in \Theta$; however, forming $P(\theta \mid \xi)$ requires normalizing over all known intents, after which $P(\theta^* \mid \xi)$ can still be high unless the user happened to operate in a way that appears equally unlikely under all known intents.

We analyzed this phenomenon by tracking a reference trajectory for the Unknown Goal task which moves optimally towards the unknown goal (see Fig. 8.2 for the task layout). We compared the performances of the distance and confidence arbitration methods, as well as a belief-based method which sets $\alpha = (P(\theta^* \mid \xi)|\Theta| - 1)/(|\Theta| - 1)$ (chosen so that $\alpha = 0$ when $P(\theta^* \mid \xi) = 1/|\Theta|$, $\alpha = 1$ when $P(\theta^* \mid \xi) = 1$). In Fig. 8.3, the confidence in each goal stays low enough that the robot would have left the user in full control; meanwhile, the relatively higher likelihood of one goal causes the belief $P(\theta^* \mid \xi)$ to quickly go to 1 and thus set the user's control authority to 0 under the belief-based arbitration scheme.

We examined one belief-based arbitration method here, but since $P(\theta^* \mid \xi)$ rapidly goes to 1, any other arbitration that is a function of the belief $P(\theta \mid \xi)$ would similarly try to assist for the wrong goal, motivating our choice of the simpler but more robust distance-based arbitration baseline.

8.2.3 Well-aligned Tasks

Fig. 8.2 (top) showcases the results of our experiment for the Known Goal task. Looking at the confidence plot, we see that $\hat{\beta}_\theta$ increases with time for the correct green goal, while it remains low for the alternate known purple goal. In the arbitration plot, as $\hat{\beta}_{\theta^*}$ increases,

α gradually decreases, reflecting that the robot takes more control authority only as it becomes more confident that the person's intent is indeed θ^*. Similarly, since there is no misalignment, PBA arbitration steadily decreases the human's contribution to the final control. Both methods result in smooth trajectories which go to the correct goal location.

8.2.4 Misaligned Tasks

CASA distinguishes itself in how it handles misaligned tasks. During the Unknown Goal task, in Fig. 8.2 (middle), CASA before learning estimates low $\hat{\beta}_\theta$ for both goals, since neither goal explains the person's motion moving towards the red goal. The estimated $\hat{\beta}_\theta$ is slightly higher for the green goal than for the purple one because it is closer to the user's input; however, neither are high enough to warrant an arbitration α below 1, and thus the robot receives no control. In Fig. 8.2 (bottom), we observe almost identical behavior before learning for the Unknown Skill task: the known intents do not match the user's behavior, and thus the user is given full control and completes the task.

This contrasts PBA, which, for both Unknown Goal and Unknown Skill, predicts the green goal as the intent. Since in both cases the user's desired trajectory passes near the green goal, PBA erroneously takes control and moves the user towards it, requiring the human to counteract the robot's controls to try to accomplish the task.

In the middle plots for each of the misaligned tasks, we observe for CASA after learning, the newly-learned intents receive confidence estimates which increase as the robot is able to observe the user, and thus CASA contributes more to the control as it becomes confident.

8.3 User Study: Confidence-Aware Assistance

We present our user study testing how well CASA can assist non-expert users.

8.3.1 Experimental Design

Due to the COVID-19 pandemic, we were unable to perform an in-person user study with a physical robot. Instead, as described in Sec. 8.2, we replicated our lab set-up in a pybullet simulator [74] in which users can teleoperate a 7 DoF JACO robotic arm using keyboard inputs (Fig. 8.2).

We split the study into four phases: (1) familiarization, (2) no misalignment, (3) misalignment before learning, and (4) misalignment after learning. First, we introduce the user to the simulation interface by asking them to perform a familiarization task. In the next phase, we tested the Known Goal task. In the third phase, we tested the two misaligned tasks, Unknown Goal and Unknown Skill, then asked participants to provide 5 demonstrations for each intent. Finally, in the fourth phase, we retested the misaligned tasks using reward functions learned from the demonstrations.

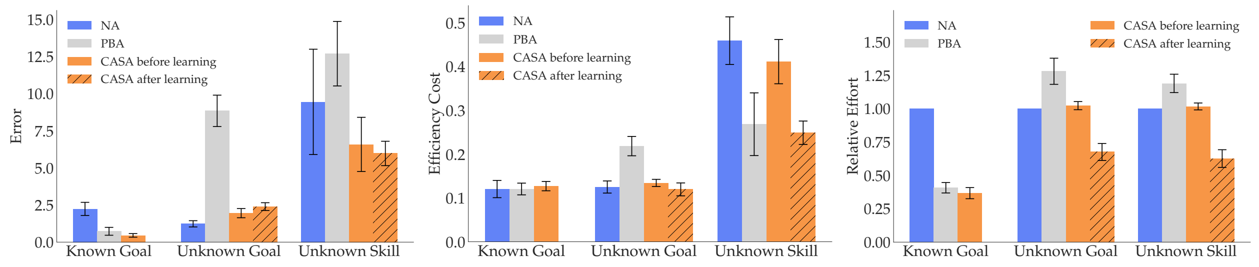


Figure 8.4: Our user study objective metrics. We measured error with respect to an intended trajectory (left), smoothness of the executed trajectory (middle), and effort relative to direct teleoperation (right).

Independent Variables: For each experiment, we manipulate the *assistance method* with three levels: no assistance (NA), policy blending assistance (PBA) [89], and Confidence-Aware Shared Autonomy (CASA). For Unknown Goal and Unknown Skill, we compared our method before and after learning new intents against the NA and PBA baselines.

Dependent Measures: Before each task, we displayed an exemplary reference trajectory to help participants understand their objective. As such, for our objective metrics, we measured *Error* as the sum of squared differences between the intended and executed trajectories, *Efficiency reward* as the sum of squared velocities across the executed trajectory, and *Effort* as the number of keys pressed. To assess the users' interaction experience, we administered a subjective 7-point Likert scale survey, asking the participants three questions: (1) if they felt the robot understood how they wanted the task done, (2) if the robot made the interaction more effortless, and (3) if the assistance provided was useful.

Participants: We used a within-subjects design and counterbalanced the order of the assistance methods. We recruited 11 users (10 male, aged 20-30) from the campus community, most of whom had technical background.

Hypotheses:
H1: If there is no misalignment, assisting with CASA is not inferior to assisting with PBA, and is superior to NA.
H2: If there is misalignment, assisting with CASA before learning is more accurate, efficient, and effortless than with PBA and not inferior to NA.
H3: If there is misalignment, assisting with CASA after learning is more accurate, efficient, and effortless than NA.
H4: If there is misalignment, participants will believe the robot understood what they want, feel less interaction effort, and find the assistance more useful with CASA after learning than with any other baseline.

8.3.2 Analysis

Objective. Fig. 8.4 summarizes our main findings. For Known Goal, which is well-aligned, CASA does no worse than PBA and better that NA in terms of relative effort and

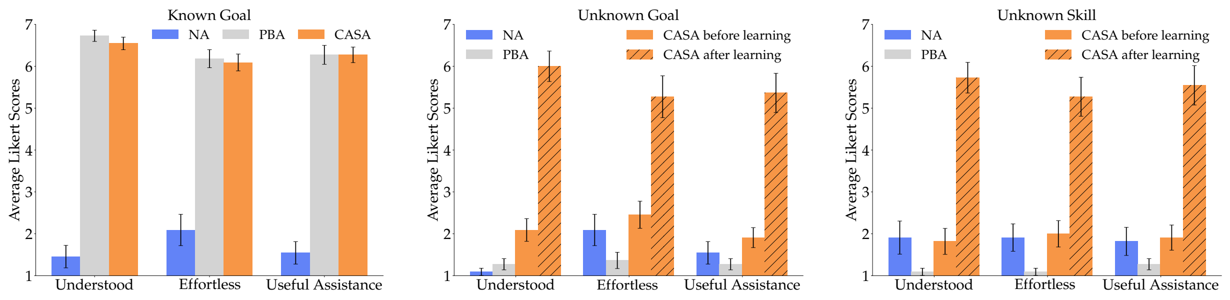


Figure 8.5: Subjective user study results. When there is no misalignment (left), our method is not inferior to PBA; when there is misalignment (center, right), participants prefer CASA after learning a new intent.

error. We confirmed this by running an ANOVA, finding a significant main effect for the method ($F(2,30) = 104.93, p < .0001$ for effort; $F(2,30) = 8.93, p = .0009$ for error). In post-hoc testing, a Tukey HSD test revealed that CASA is significantly better than NA ($p < .0001$ for effort, $p = .0013$ for error). We also performed a non-inferiority test [173], and obtained that CASA is non-inferior to PBA within a margin of 0.065 for effort, 0.025 for efficiency, and 0.26 for error. These findings are in line with H1 and were expected, since the robot should have no problem handling known intents.

For the two misaligned tasks, we first ran an ANOVA with the method as a factor, and the task as a covariate, and found a significant main effect ($F(2,62) = 11.8255, p < .0001$ for effort; $F(2,62) = 6.119, p = .0038$ for error). A Tukey HSD revealed that CASA is significantly better than PBA ($p = .0005$ for effort, $p = .005$ for error). We also ran a non-inferiority test, and obtained that CASA is non-inferior to NA within a margin of 0.035 for effort, 0.02 for efficiency, and 1.4 for error for Unknown Goal, and 0.03 for effort, 0.09 for efficiency, and 4.5 for error for Unknown Skill. For both unknown tasks, CASA before learning is essentially indistinguishable from NA since a low $\hat{\beta}_{\theta^*}$ would make the robot rely on direct teleoperation. Both the figure and our statistical tests confirm H2, which speaks for the consequences of confidently assisting for the wrong intent.

For efficiency reward, we did not find an effect, possibly because Fig. 8.4 shows that PBA is more efficient for the Unknown Skill task than other methods. Anecdotally, PBA forced users to an incorrect goal thus preventing them from pouring, which explains the lower efficiency reward. By having a high arbitration for the wrong intent, PBA can cause a smooth trajectory, since it lowers the control authority of the possibly-noisy human inputs. However, this trajectory does not accomplish the task. When running an ANOVA for each of the tasks separately, we found a significant main effect for the method for Unknown Goal ($F(2,30) = 9.66, p = .0006$), and a post-hoc Tukey HSD revealed CASA is significantly better than PBA ($p = .0032$), further confirming H2.

Lastly, we looked at the performance with CASA after learning the new intents. For Unknown Goal, a simple task, the figure shows that CASA after learning doesn't improve efficiency and error, but it does reduce relative effort when compared to NA. For Unknown

Skill, a more complex task, CASA after learning outperforms NA. This is confirmed by an ANOVA with the method (NA, CASA after learning) as the factor, where we found a significant main effect ($F(1, 41) = 53.60, p < .0001$ for effort; $F(1, 641) = 8.6184, p = .0054$ for efficiency reward), supporting H3.

Subjective. We show the average Likert survey scores for each task in Fig. 8.5. In line with H1, for the Known Goal task, users thought the robot under both PBA and CASA had a good understanding of how they wanted the task to be done, made the interaction more effortless, and provided useful assistance. The results are in stark contrast to NA, which scores low on all those metrics. For Unknown Goal and Unknown Skill, all methods fare poorly on all questions except for CASA after learning, supporting H4.

8.4 Discussion

In this chapter, we formalized a confidence-aware shared autonomy process where the robot can adjust its assistance based on how confident it is in its prediction of the human intent. We introduced an approximate solution for estimating this confidence, and demonstrated its effectiveness in adjusting arbitration when the robot's intent set is misaligned and enabling continual learning of new intents.

While our confidence estimates tolerated some degree of suboptimal user control, an extremely noisy operator attempting a known intent might instead appear to be performing a novel intent. Moreover, due to COVID, we ran our experiments in a simulator, which does not replicate the difficulty inherent in teleoperating a real manipulator via a joystick interface. Despite these limitations, we are encouraged to see robots have a more principled and robust way to arbitrate shared autonomy, as well as decide when they need to learn more to be better teammates. We look forward to applications of our confidence-based ideas beyond manipulation robots, to semi-autonomous vehicles, quadcopter control, or any other shared autonomy scenarios.

Part IV

Using Aligned Representations to Interpret Human Behavior

Thus far, this thesis has showcased several implications and benefits of aligning human and robot representations, from more robust and generalizable reward learning, to personalized motion planning, to introspective robots that can autonomously detect misalignment. However, once we agree that representations should not be just functional but also human-aligned, we can also fundamentally rethink how robots *model* people internally, and, thus in turn, interpret their behavior. In Part IV, we observe that when humans make decisions, their representation affects how they view the options amongst which they are choosing. For example, although there are many paths around an obstacle, humans may group them into “left” and “right” when deciding between them. Instead of seeing human decisions as a choice from a set of behaviors, as current models do, we develop a novel computational model of human decision-making that *reinterprets the available choices from the lens of the human-aligned representation*. We demonstrate that this better reflects how humans make decisions, and that robots make better inferences if they interpret human input as such. Crucially, our model has wide-reaching impacts beyond robotics, to artificial intelligence, econometrics, and cognitive science.

Chapter 9

A Novel Human Decision-Making Model

This chapter is based on the paper "LESS is More: Rethinking Probabilistic Models of Human Behavior" [44] written with Dexter Scobee, Jaime Fisac, Shankar Sastry, and Anca Dragan.

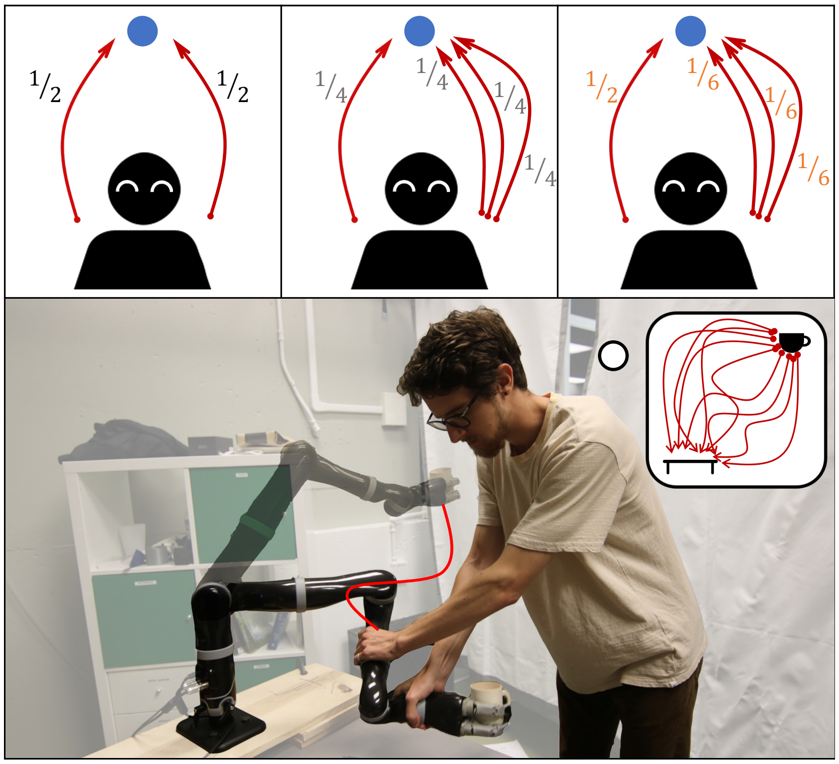


Figure 9.1: (Top) Contrary to Boltzmann (gray), when adding more options on the right LESS (orange) does not drastically reduce the probability of selecting the left option. (Bottom) We test LESS on learning from user demonstrations for a 7DOF JACO arm.

What we do depends on our intent – our goals and preferences. When robots collaborate with us, they need to observe our behavior and infer our intent from it, so that they can help us achieve it. They also need to anticipate or predict our future behavior given what they have inferred, so that they can seamlessly coordinate their behavior with ours. Both inference and prediction require a model of human behavior conditioned on intent.

A very popular such model is Boltzmann rationality [27, 282]. It formalizes intent via a reward function, and models the human as selecting trajectories in proportion to their (exponentiated) reward. Boltzmann rationality has seen great successes in a variety of robotic domains, from mobile robots [164, 280, 131, 312, 226] to autonomous cars [311, 291, 157] to manipulation [150, 43, 96, 198, 199], in both inference [175, 311, 150, 164, 96, 232, 7, 280, 131] and prediction [157, 198, 312, 199, 226].

Despite its widespread use, Boltzmann predictions are not always the most natural. At the core of the Boltzmann model is the view that behavior is a choice among available alternatives; the probability of any trajectory thus heavily depends on the available alternatives. This has some unforeseen side-effects. One of the simplest examples is at the top of Fig. 9.1. Imagine first that there are two possible trajectories to a goal, left and right, both equally good. Boltzmann would predict a .5 probability of choosing to go to the left. Next, imagine that we change the set of alternatives: we add two similar trajectories to the right. Just because there are more options to go to the right, Boltzmann now predicts a higher probability that you will decide to do so: for these four equally good trajectories, Boltzmann assigns .25 probability each, and estimates going left with only .25 probability instead of .5 as before. Should this change in alternatives – the addition of similar options to go to the right – really be reducing the prediction that you will go left by *that* much?

This example seems artificial – when do we have a) a group of similar trajectories, and b) an imbalance in the number of similar trajectories for each option, so that Boltzmann shows this side-effect? Unfortunately, it is quite representative of real-world trajectory spaces. Spaces of trajectories are *continuous and bounded*, so they naturally contain a *continuum* of alternatives of varying similarity to each other, just like the right-side trajectories in our example. Further, trajectories will have varying amounts of similarity to the rest of the space: just like our left-side trajectory was dissimilar from the other alternatives, in the real world, trajectories closer to joint limits or that squeeze in between two nearby obstacles will be dissimilar from the rest of the trajectory space. In other words, the features that the space exhibits affects how similar certain trajectories are to others.

Unfortunately, the Boltzmann model was not designed to handle this. It has its roots in the Luce axiom of choice from econometrics and mathematical psychology [195, 194], which models decisions among *discrete and different* options. When we move to trajectory spaces, the options now are all connected to some degree via their feature representation:

Our insight is that we need to rethink how to generalize the Luce axiom to trajectory spaces, and account for how representation ***similarity*** *in trajectories should influence their probability.*

We take a first step towards this goal by introducing an alternative to the Boltzmann model that accounts not just for the reward of each trajectory, but also for the feature-space similarity each trajectory has with all other alternatives. That is, we use the human-aligned representation features to account for the similarity amongst trajectory options. We name our model LESS: Limiting Errors due to Similar Selections. We start by testing that our model does better at predicting human decision (Sec. 9.2), and then move on to analyze its implications for inference. We first conduct experiments in simulation, with ground truth reward functions, to show that we can make more accurate inferences using our model (Sec. 9.3). Finally, we test inference on real manipulation tasks with a 7DOF arm, where we learn from user demonstrations (Sec. 9.4)– though we no longer have ground truth, we show that we can improve the robustness of the inference if we use LESS.

9.1 Approach: The LESS Human Decision Model

Motivated by human prediction and reward inference for robotics, we seek an improved human behavior model, designed for *trajectory* spaces rather than abstract discrete decisions. To develop this theory, we first turn to the literature on human decision making.

9.1.1 Background

9.1.1.1 Human Decision Making. One of the preeminent theories of human decision making in mathematical psychology is based on Luce's axiom of choice [194, 195]. Here, we consider a set of options $\mathcal{O}$, and we seek t quantify the likelihood that a human will select any particular option $o \in \mathcal{O}$. The desirability of each option can be modeled by a function $v : \mathcal{O} \to \mathbb{R}^+$, where v produces higher values for more desirable options. As a consequence of Luce's choice axiom, the probability of selecting an option o is given by

$$P(o) = \frac{v(o)}{\sum_{\bar{o}\in\mathcal{O}} v(\bar{o})} \quad . \tag{9.1}$$

If we further assume that each option o has some underlying reward $R(o) \in \mathbb{R}$, and we allow desirability to be an exponential function of this reward, then we recover the Luce-Shepard choice rule [261]:

$$P(o) = \frac{e^{R(o)}}{\sum_{\bar{o}\in\mathcal{O}} e^{R(\bar{o})}} \quad . \tag{9.2}$$

When the options being chosen by the human are trajectories $\xi \in \Xi$, i.e. sequences of (potentially continuous-valued) actions, we refer to (9.2) as the Boltzmann model of noisily-rational behavior [282, 27]. The reward R is typically a function of a feature vector $\phi : \Xi \to \mathbb{R}^k$, giving the probability density p over continuous Ξ as

$$p(\xi) = \frac{e^{R(\phi(\xi))}}{\int_{\Xi} e^{R(\phi(\bar{\xi}))} d\bar{\xi}} \quad . \tag{9.3}$$

9.1.1.2 Handling Duplicates. Since the introduction of the Luce axiom, related works [79, 114] have pointed out its *duplicates problem*, where inserting a duplicate of any option o into $\mathcal{O}$ has an undue influence on selection probabilities. To address this, various extensions of the Luce model have been proposed which attempt to group together identical or similar options [32, 283]. Further extending these ideas, Gul, Natenzon, and Pesendorfer [114] recently introduced the *attribute rule*, which reinterprets options as bundles of attributes but maintains Luce's idea that choice is governed by desirability values.

Analogous to [114], let $\mathcal{X}$ be the set of all attributes, let $\mathcal{X}_o \subseteq \mathcal{X}$ be the set of attributes belonging to o, and let $\mathcal{X}_{\mathcal{O}} \subseteq \mathcal{X}$ be the set of attributes which belong to at least one option $o \in \mathcal{O}$. Define an *attribute value*, $w : \mathcal{X} \to \mathbb{R}^+$, that maps attributes to their desirability, and an *attribute intensity*, $s : \mathcal{X} \times \mathcal{O} \to \mathbb{N}$, that maps pairs of attributes and options to natural numbers, usually 0 or 1, to indicate the degree to which an attribute is expressed. For instance, an attribute could be the property "green" and $s(\text{"green"}, o)$ could return 1 if option o, say one of a set of cars, is green, and 0 otherwise.

According to the attribute rule, the probability of choosing o is

$$P(o) = \sum_{x \in \mathcal{X}_o} \frac{w(x)}{\sum_{\bar{x} \in \mathcal{X}_{\mathcal{O}}} w(\bar{x})} \cdot \frac{s(x, o)}{\sum_{\bar{o} \in \mathcal{O}} s(x, \bar{o})} \quad , \tag{9.4}$$

which describes a process where the human first chooses an attribute $x \in \mathcal{X}_{\mathcal{O}}$ according to a Luce-like rule, then an option $o \in \mathcal{O}$ with that attribute according to another Luce-like rule. Note that (9.4) reduces to (9.1) if no pair of options in $\mathcal{O}$ shares any attributes; for example, if each o has a single unique attribute, the first sum in (9.4) disappears, and the second fraction evaluates to 1. In this work, we want to take advantage of the attribute rule's graceful handling of duplicates while extending its functionality to trajectories with continuous-valued features and not only categorical attributes.

9.1.2 The LESS Human Decision Model

In this paper, we take inspiration from the attribute rule to derive a novel model of human decision making in continuous spaces. Key to our approach is introducing a similarity measure on trajectories. This could be directly in the trajectory space, but more generally it is in *feature* space, where features could, in one extreme, be the trajectory itself. We first instantiate the attribute rule with features as the attributes, and then soften it to account for feature similarity. Indeed, the Boltzmann rationality model given by (9.3) already assigns selection probabilities based only on trajectory features, so we look to modify the decision space to depend directly on features as well.

9.1.2.1 Accounting for Trajectories with Identical Features. We derive our model by starting from (9.4) and defining the set of attributes to be Φ, the set of all possible feature vectors. Accordingly, the set of attributes that belong to ξ is a single element $\Phi_\xi = \{\phi(\xi)\}$, and the attributes represented in a set $\Xi' \subseteq \Xi$ are $\Phi_{\Xi'} = \{\phi(\xi') \mid \xi' \in \Xi'\}$. Combining this convention with the reward model (9.3), the modified attribute rule for trajectories over a finite subset $\Xi_f \subset \Xi$ becomes

$$P(\xi) = \frac{e^{R(\phi(\xi))}}{\sum_{\bar{\phi}\in\Phi_{\Xi_f}} e^{R(\bar{\phi})}} \cdot \frac{s(\phi(\xi),\xi)}{\sum_{\bar{\xi}\in\Xi_f} s(\phi(\xi),\bar{\xi})} \ . \tag{9.5}$$

In the original attribute rule, the attribute intensity s mapped to natural numbers. A convenient mapping here would be to use s as an indicator function, where $s(x,\xi)$ evaluates to 1 only if $x = \phi(\xi)$. With this formulation, *if* all trajectories have a unique feature vector, then the rightmost term of (9.5) is identically 1 and we recover the Boltzmann model (9.3), as applied to a finite sample of trajectories Ξ_f. If, on the other hand, multiple trajectories share the exact same feature vector, then they will effectively be considered as a single option, and the selection probability will be distributed equally among them. This effect is desirable: since the features $\phi(\xi)$ capture all the relevant inputs to the reward, trajectories with the same features should be considered practically equivalent.

9.1.2.2 Softening to Feature Similarity. We suggest that such a notion of attribute intensity is too stringent for continuous spaces, and we redefine s to be a soft *similarity metric* $s : \Phi \times \Xi \longrightarrow \mathbb{R}^+$, which should be symmetric ($s(\phi(\xi),\bar{\xi}) = s(\phi(\bar{\xi}),\xi)$) and positive semidefinite ($s(x,\xi) \geq 0$), with $s(\phi(\xi),\xi) = \max_{x\in\Phi,\bar{\xi}\in\Xi} s(x,\bar{\xi})$ for all $\xi \in \Xi$.

Using this redefined similarity metric s, we extend (9.5) to be a probability density on the continuous trajectory space Ξ, as in (9.3):

$$p(\xi) = \frac{\frac{e^{R(\phi(\xi))}}{\int_\Xi s(\phi(\xi),\bar{\xi})\, d\bar{\xi}}}{\int_\Xi \frac{e^{R(\phi(\hat{\xi}))}}{\int_\Xi s(\phi(\hat{\xi}),\bar{\xi})\, d\bar{\xi}}\, d\hat{\xi}} \propto \frac{e^{R(\phi(\xi))}}{\int_\Xi s(\phi(\xi),\bar{\xi})\, d\bar{\xi}} \ , \tag{9.6}$$

where $s(\phi(\xi),\xi)$ and the integral over Φ_Ξ are omitted because they are constant over Ξ and cancel out during normalization.

Under this new formulation, the likelihood of selecting a trajectory is inversely proportional to its feature-space similarity with other trajectories. This de-weighting of trajectories that are similar to others is precisely the effect we seek, and we adopt the probability given by (9.6) as our LESS model of human decision making.

9.1.3 Similarity as Density

The main innovation that differentiates our model from previously proposed rules is the use of a similarity metric that reweights trajectory likelihoods based on the presence of other trajectories that are nearby in feature space. The integral of this similarity over trajectories, the denominator of (9.6), is akin to a measure of trajectory density in feature space. We estimate similarity as a density by selecting our similarity metric as a kernel function and performing Kernel Density Estimation (KDE). There are many choices of kernel functions, each parametrized by some notion of bandwidth. In our experiments, we used a radial basis function, which peaks when $x = \phi(\xi)$, then exponentially decreases the farther away x and $\phi(\xi)$ are from one another in feature space:

$$s(x, \xi) = \left(\frac{1}{\sigma\sqrt{2\pi}}\right) \exp\left(-\frac{\|x - \phi(\xi)\|^2}{2\sigma^2}\right), \tag{9.7}$$

where the bandwidth σ is an important parameter that dictates, for a given feature difference between two trajectories, how much that difference affects the ultimate similarity evaluation. Higher σ means a higher bandwidth and makes everything look more similar.

We find an optimal bandwidth σ^* automatically by using a finite set of samples $\Xi_f \subset \Xi$ and maximizing the sum of the log of their summed similarities, which is equivalent to maximizing their likelihood under a probability density estimate produced by KDE:

$$\sigma^* = \arg\max_{\sigma \in \mathbb{R}} \sum_{\xi \in \Xi_f} \log\left(\sum_{\bar{\xi} \in \Xi_f} s(\phi(\xi), \bar{\xi})\right) . \tag{9.8}$$

9.1.4 Inference and Prediction with LESS

Let $\theta \in \Theta$ parametrize the reward function R. To predict what the human will do given a belief $b(\theta)$, we marginalize over θ:

$$p(\xi) = \int_{\Theta} b(\theta) p(\xi|\theta) d\theta \ , \tag{9.9}$$

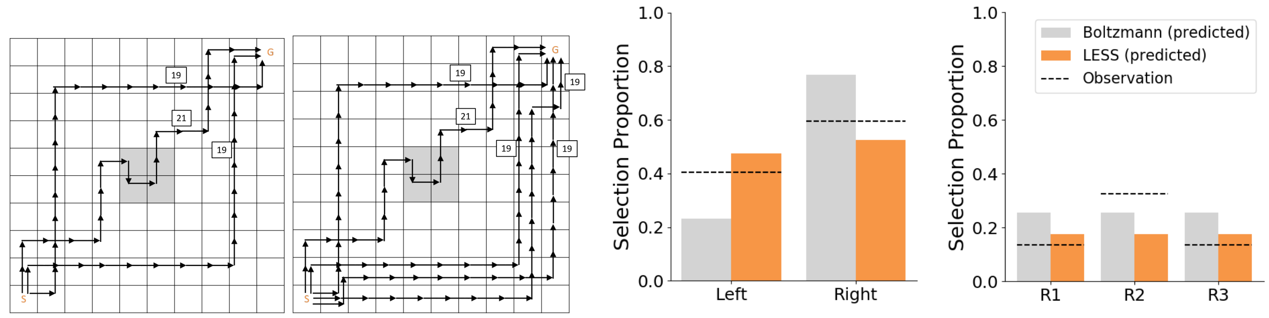
with $p(\xi|\theta)$ given by (9.6). To perform inference over θ given a human trajectory, we update our belief using Bayesian inference:

$$b'(\theta) = \frac{b(\theta)p(\xi|\theta)}{\int_{\Theta} b(\bar{\theta})p(\xi|\bar{\theta})d\bar{\theta}} \ . \tag{9.10}$$

In practice, calculating the integrals in the denominators of (9.10) and (9.6) can be intractable, so we use a discretized set of θ parameters and finite trajectory sample sets in our experiments. The specific sampling of the trajectory choice space can significantly impact inference, and we explore its implications in Sec. 9.4.

9.2 User Study: LESS as a Human Decision Model

We start by testing the hypothesis that LESS is a better model for human decision making than the standard Boltzmann model.



(a) Control trial (b) Experimental trial (c) Left and Right distributions. (d) Distributions within Right

Figure 9.2: The human decision model experiment. a and b show the trajectories used for the two trials. In c, LESS predictions more closely match the observed Left-Right distribution. In d, both models miss users' slight preference for R2 (the trajectory which visits the most states in the rightmost column in b).

9.2.1 Human Decision Model Experiment Design

We design a browser-based user study in which we ask participants to make behavior decisions, and measure which model best characterizes these decisions. We select a simple navigation task as our domain, where different behaviors correspond to different ways of traversing the grid from start to goal, as shown in Fig. 9.2.

Main Design Idea. The key difficulty in designing such a study is that both models require access to a ground truth reward function, i.e. user preferences over trajectories. Even though we can provide participants with some criteria – in our case optimizing for path length while avoiding the obstacle –, this does not mean our criteria are the only ones they care about. For instance, people might implicitly prefer trajectories that go closer to or further from the obstacle, or that go around the obstacle to the left or right.

Our design idea is to introduce a control trial in which we gather data about *relative* preferences among two *dissimilar* options: left and right. These relative preferences then enables us to make predictions, under each model, about the experimental trial, where we add trajectories similar to the option on the right.

For the control trial, participants saw the grid world shown in Fig. 9.2a with one obstacle in the middle and three trajectories travelling between the start and goal. Two of the trajectories traversed an equal amount of tiles (optimal) and were symmetric along the diagonal of the grid (left and right), and a third trajectory went through the obstacle and visited more tiles than the others (not optimal). We were only interested in what specific optimal trajectory people chose (Left versus Right), and we used the third suboptimal trajectory as an attention test to check if subjects had paid attention to the instructions. We chose the two optimal trajectories to be symmetric and of the same color to reduce possible confounds, such as bias people might have for extraneous features like number of turns, distance from obstacle, color, etc.

For the experimental trial, shown in Fig. 9.2b, we had the same setup as in the control, with the addition of two other optimal trajectories on the right. They had the same color, number of turns, and number of tiles traversed as the original right-side trajectory. In this setup, there were two visible clusters of options: one trajectory on the left, and three clustered on the right, which we denote as the Left and Right groups, respectively.

Manipulated Variables. We manipulated the model used for decision-making in the experimental trial to be Boltzmann vs. LESS. Having access to the ratio λ that participants chose the left trajectory over the right in the control trial means that regardless of their reward function $R(\xi)$, $e^{R(\xi_{left})} = \lambda e^{R(\xi_{right})}$, according to (9.3). This enables us to make predictions using both models as a function of λ for the experimental trial, despite not knowing R itself. For these computations, we assumed that all trajectories in the Right group had the same reward, that the reward of trajectories in the Left and Right groups would be equal to those estimated from the control trial, and (for LESS) that the Left trajectory had density one while the Right trajectories had density three.

Under the Boltzmann model, the addition of two trajectories similar to the one on the right decreases the probability that the left trajectory is chosen. This is most obvious when $\lambda = 1$, i.e. if users liked both trajectories equally – then, $P(\xi_{\text{left}})$ would go from .5 all the way down to .25, as there are now 4 good options. On the other hand, LESS accounts for the similarity of the trajectories on the right and keeps $P(\xi_{\text{left}})$ closer to the control value.

Dependent Measures. We report the selection proportion of each trajectory in the experimental trial, and compute agreement between each model and the users' decisions.

Subject Allocation. We recruited 80 participants (24 female, 56 male, with ages between 18 and 65) from Amazon Mechanical Turk (AMT) using the psiTurk experimental framework [117]. We excluded 3 participants for failing our attention test. All participants were from the United States and had a minimum approval rating of 95%. The treatment trial was assigned between-subjects: participants saw only one of the sets of trajectory options.

Hypotheses.
H1: For the experimental trial, the Boltzmann proportion prediction is significantly different from the observed proportion.
H2: For the experimental trial, the LESS proportion prediction is equivalent to the observed proportions.

9.2.2 Analysis

In the control trial, users chose the Left trajectory 47.5% of the time. Fig. 9.2 plots the observed proportions for the experimental trial, along with each model's predictions. The experimental trial resulted in an observed probability of .41 for the Left trajectory, whereas Boltzmann predicts .23 and LESS predicts .475. The models both predict a uniform distribution among the Right trajectories.

We performed a chi-square test of goodness of fit to see if the observed distribution of left vs. right from the experimental group differed from the predicted distributions. In line with our hypotheses, we found a significant difference between the observed values and the Boltzmann prediction ($X^2(1, N = 37) = 6.27$, $p < 0.05$), and no significant difference between the observations and the LESS prediction ($X^2(1, N = 37) = 0.72$, $p = 0.4$).

To test for equivalence, we performed an equivalence test for multinomial distributions as described by Wellek [288]. This test evaluates the null hypothesis that the Euclidean distance between the multinomial distribution and a reference is greater than some ϵ (where the distance is computed by taking each distribution to be a vector in $[0, 1]^k$, where k is the number of trajectories represented by the distribution). We do not have an a priori estimate for which values of ϵ are practically insignificant in this vector space of probability distributions, so we instead invert the test to find the minimum ϵ for which the observed distribution matches the predicted distribution at a significance level of $\alpha = 0.05$. We found that the minimum ϵ bound for equivalence at the $\alpha = 0.05$ level was 0.22 for the LESS prediction and 0.39 for the Boltzmann prediction.

The results across all trajectories are analogous, albeit slightly weaker because users tended to favor one of the three Right trajectories more than the other two. The chi-square test revealed a significant difference with the Boltzmann predictions, $X^2(1, N = 37) = 9.72$, $p < 0.05$, but no significant difference between the observations and the LESS prediction $X^2(1, N = 37) = 5.76$, $p = 0.12$.

The equivalence test found the observed distribution matches the LESS-based predicted distribution at a significance level of $\alpha = 0.05$ when the ϵ bound is 0.29, and 0.36 for Boltzmann. Despite LESS' tighter ϵ, neither prediction aligns perfectly with the empirical data in Fig. 9.2d. This discrepancy is likely due to some unmodeled features (e.g. distance from the obstacle), which may influence participants' preferences. However, while unknown features may affect both Boltzmann's and LESS' performance, LESS still corrects Boltzmann's errors from mishandling similarity. We explore the specific effects of feature misalignment further in Sec. 9.3.3.

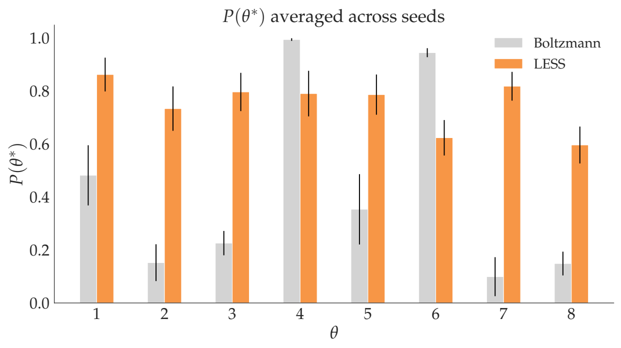
Overall, although neither model is a perfect predictor of behavior, we find that LESS is a better fit: Boltzmann is significantly different from the observed, and LESS provides a tighter equivalence bound.

9.3 Experiments: Using LESS for Robot Inference

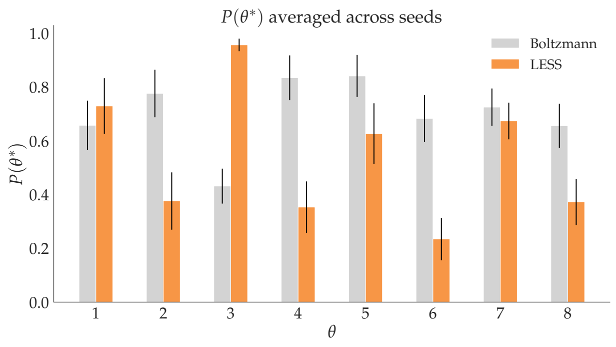
In Sec. 9.2, we provided evidence supporting that LESS can more accurately capture human decisions. This has direct implications for how robots predict behavior – increasing the model accuracy by definition increases prediction accuracy. We now hypothesize that it also has implications for how robots *infer* human preferences from behavior: using a higher accuracy model when performing inference leads to more accurate inference.

9.3.1 Boltzmann and LESS Inference Comparison

We first design an experiment to test that if people do act according to the LESS distribution, modeling them as such leads to better inference than modeling them via



(a) $TruePosterior$ for LESS sampling model.



(b) $TruePosterior$ for Boltzmann sampling model.

Figure 9.3: *TruePosterior* results for the inference comparison experiment in Sec. 9.3.1. Legends indicate which inference method was employed for those results. We found a significant interaction effect between sampling method and inference method, which can be seen in the change of relative performance for LESS and Boltzmann between a and b.

Boltzmann. To control for potential confounds, we also verify the opposite: if instead people acted according to Boltzmann (which Sec. 9.2 does not support), then modeling them as Boltzmann would instead be better for inference.

We created a grid world environment with two objects, where humans have to teach a robot to navigate from a start to a goal and learn preferences for whether to stay close or far from the objects. We simulated hypothetical human demonstrations Ξ_D by sampling trajectories according to LESS and Boltzmann. To do so, we fixed a particular objective θ^* and a confidence parameter β, and randomly chose trajectories according to probabilities given by either (9.6), for LESS, or (9.3), for Boltzmann. We then utilized these trajectories as "human" demonstrations and performed inference using either Boltzmann or LESS as the underlying choice model. Our goal was to analyze how each model's inference quality depends on the sampling model used across a range of objectives θ^*.

Manipulated Variables. We used a 2-by-2 factorial design. We manipulated the *sampling model* with two levels, Boltzmann and LESS, and the *inference model*, Boltzmann and LESS.

Other Variables. For variation, we tested inference quality across eight different θ^* values, and used 150 random seeds for sampling demonstrations. For a given sampling method, the combination of a θ^* and a seed determine the demonstration set that the inference will use. Therefore, we generated 1200 demonstration sets for each sampling method.

Dependent Measures. To analyze each model's inference quality, we use two metrics:
Accuracy of a-posteriori inference: once we obtain a posterior probability induced by the sampled Ξ_D, we verify that the maximum a-posteriori θ^{MAP} matches the original θ^*. Thus, we define a binary variable that takes value 1 if they match and 0 otherwise:

$$TrueMatch = \mathbb{1}\{\theta^{MAP} = \theta^*\}.$$

Magnitude of posterior θ^* probability: this gives a softened, continuous indication of inference performance by capturing the posterior probability mass assigned to the correct θ^*:

$$TruePosterior = P(\theta^* \mid \Xi_D).$$

Hypotheses.
H3: When human input is generated using LESS, inference quality is significantly higher with LESS than with Boltzmann.
H4: When human input is generated using Boltzmann, inference quality is significantly higher with Boltzmann than with LESS.

Analysis. Fig. 9.3 summarizes the results by showing how *TruePosterior* varies by inference method. To analyze these results, we ran a factorial repeated measures ANOVA. We found a significant interaction effect between the sampling and inference methods ($F(1, 1199) = 965.06, p < 0.001$), which can be seen with the change in relative performance of Boltzmann and LESS from Fig. 9.3a to Fig. 9.3b. A factorial logistic regression for the *TrueMatch* results also revealed a significant interaction between sampling method and inference method ($p < .001$). In post-hoc testing, a Tukey HSD test revealed that *TruePosterior* was significantly higher when the inference method matched the sampling method ($p < .001$ for both), and logistic regressions similarly showed that the probability of *TrueMatch* $= 1$ is greater when sampling and inference agree ($p < .001$ for both).

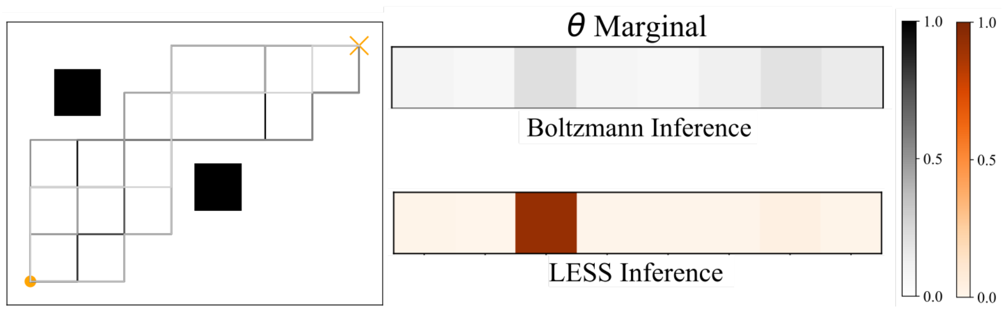
These results strongly support both H3 and H4, as they reveal that inference performance is superior when the inference method agrees with the sampling method. Given that the experiment in Sec. 9.2 suggests that LESS can be a better model of human sampling behavior, these results provide evidence that using LESS-based inference could give better performance when learning from humans.

9.3.2 Qualitative Analysis of LESS Inference

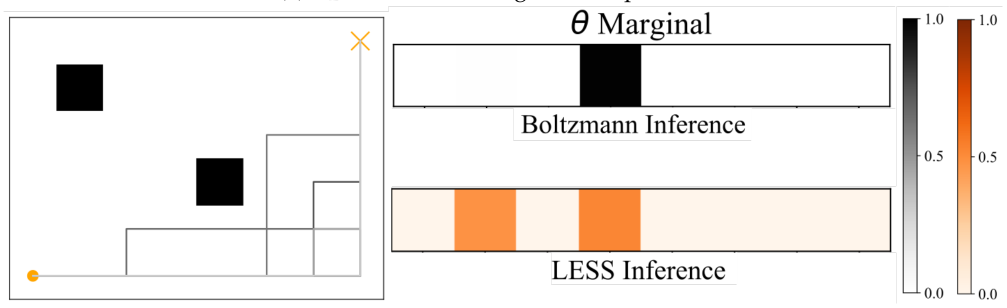
Based on what we have seen thus far, LESS clearly leads to different robot inferences. In this section we provide some qualitative intuition about what contributes to this difference.

The important change from Boltzmann to LESS is the *strength* of the inference as a function of the feature *density* at the demonstrated trajectory. If a demonstrated trajectory lies in a high-density area, i.e. its features are similar to those of many other possible trajectories, Boltzmann inference will *under-learn*. This is because there are many high-reward alternatives in the normalizer of (9.3), which lowers the probability of the demonstration. For the analogous reason, if a demonstration lies in a low-density area, Boltzmann inference will *over-learn*. Because LESS weighs each trajectory ξ by the inverse of the density at its location in feature space $\phi(\xi)$, the resulting weighted density will be approximately uniform, not allowing the feature density to influence the strength of the inference: other options with similar features do not skew the probability as much anymore.

To visualize this, we chose two sets of demonstrations from the previous experiment. One set, Ξ_B, comes from one of the ground truth rewards for which Boltzmann performed



(a) Ξ_L and the resulting inferred posterior



(b) Ξ_B and the resulting inferred posterior

Figure 9.4: Visualizations of Ξ_L and Ξ_B, and the LESS and Boltzmann inferred posteriors over θ. a: LESS learns the correct θ, whereas Boltzmann under-learns. b: Boltzmann learns the correct θ, while LESS is split between avoiding both obstacles vs. avoiding the top one but being ambivalent about the bottom one.

better (θ_4 in Fig. 9.3a). The other set, Ξ_L, comes from one for which LESS performed better (θ_3 in Fig. 9.3b). Fig. 9.4 shows the samples Ξ_L and Ξ_B, along with the inference for each model. For Ξ_L, LESS confidently identifies the ground truth, whereas Boltzmann's posterior is higher entropy. Fig. 9.5 shows that Ξ_L does fall in a high-density region, which indeed leads to Boltzmann under-learning and finding many alternative explanations.

For Ξ_B, on the other hand, something very interesting happens. Looking at where the samples lie (blue dots in Fig. 9.5), two of them are in relatively high-density areas (call them Ξ_B^{dense}), whereas the others are in a very sparse region (call them Ξ_B^{sparse}). Ξ_B^{dense} are the two with lower ϕ_2 in Fig. 9.5 (right). They correspond, in Fig. 9.4b, to the two trajectories that go closer to the bottom obstacle. To the LESS inference, which is more agnostic to the feature density, this gives evidence for two hypotheses: Ξ_B^{dense} support the hypothesis that the robot should stay far from the top obstacle, but be ambivalent about the bottom one, whereas the other trajectories, Ξ_B^{sparse}, support that the robot should stay far from both obstacles. This is why we see two hypotheses inferred by LESS in Fig. 9.4b. The Boltzmann inference, however, learns much more from the trajectories that lie in the low-density area, essentially ignoring Ξ_B^{dense}. This is what leads to the very confident inference of only one of the hypotheses. In this case, this happens to be the correct hypothesis. In general though, the opposite could have happened – had the two trajectories that go closer to the

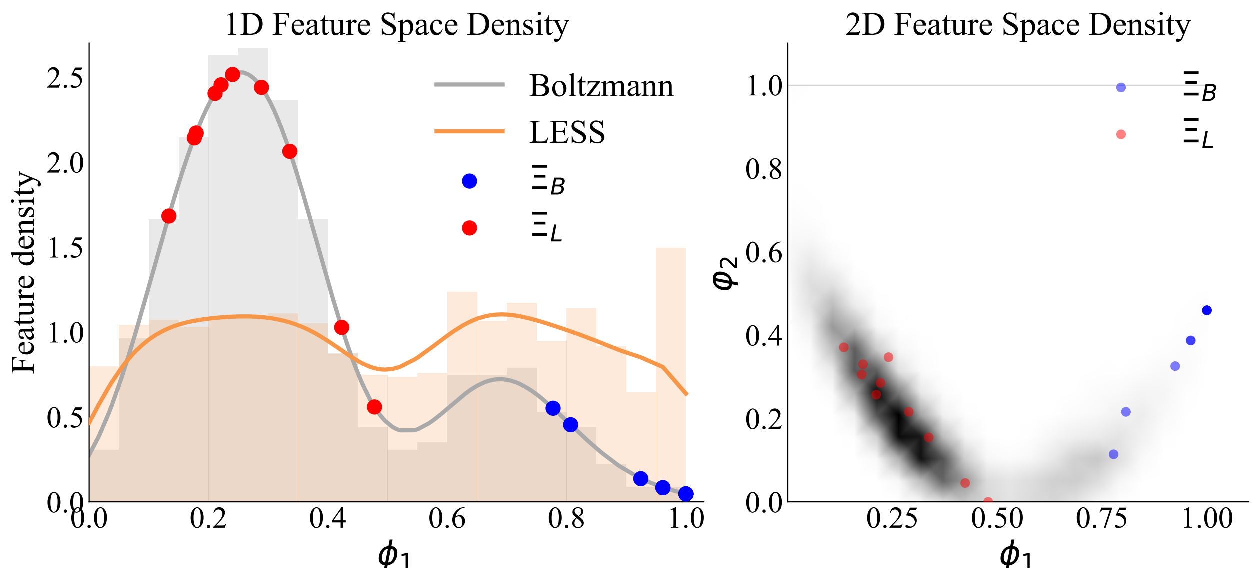


Figure 9.5: Left: actual feature density (gray), adjusted by LESS (orange). The Ξ_L points (red) are in dense areas, thus Boltzmann inference under-learns. The Ξ_B points are in sparse areas, but two of them are in a slightly more dense area, which makes Boltzmann reduce their relative influence and ignore the θ they suggest. Right: 2D density with Ξ_B, Ξ_L overlaid.

obstacle been the ones to lie in a sparse area, Boltzmann would have confidently inferred the wrong objective. In summary, Boltzmann, by being sensitive to feature densities, can under- or over-learn.

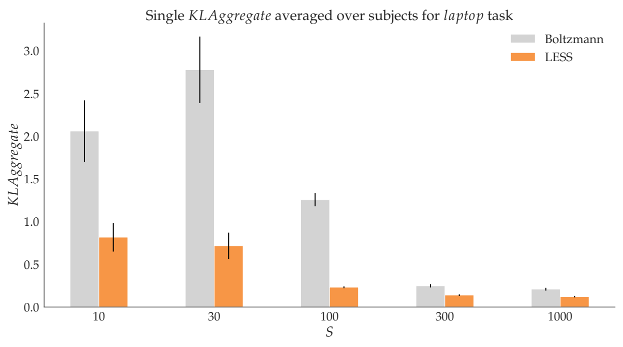
9.3.3 LESS and Feature Misalignment

LESS uses information from features to compute similarity, even when those features do not affect the reward. For example, if the reward is solely about efficiency, LESS captures that people treat "right-of-the-obstacle" options as similar. What if the robot does not have access though to these additional features?

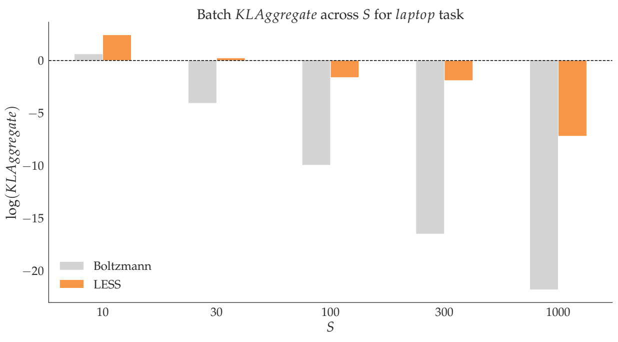
Experimental Design. We generate demonstrations using LESS, but we include two additional features: the average x and average y coordinate of the trajectory. The two new features do not influence the reward values, but they do influence the similarity metric. To induce a misalignment, the robot performing inference is unaware of these new features. For this experiment, we only manipulate the *inference model*: LESS vs. Boltzmann.

H5: When the robot's feature space is misspecified, inference quality with LESS is still superior to inference quality with Boltzmann for LESS-sampled demonstrations.

Analysis. For *TruePosterior*, we performed a one-way repeated measures ANOVA, and as hypothesized, the test revealed that LESS inference was still significantly better than Boltzmann, in spite of the feature misalignment ($p < .001$). Similarly for *TrueMatch*,



(a) *KLAggregate* for single inference comparison.



(b) log(*KLAggregate*) for batch inference comparison.

Figure 9.6: Results for the *laptop* task in the robustness analysis experiments. In a, LESS significantly outperforms Boltzmann at low sample sizes, but they converge for the largest sample sizes. For the batch inference task in b, Boltzmann outperforms LESS at the lowest sample size, but the two methods converge towards zero as sample size increase.

a logistic regression revealed that the odds of having *TrueMatch* = 1 were significantly greater when using LESS ($p < .001$), strongly supporting our hypothesis.

We take this result with a grain of salt: in the worst case, if an unspecified feature completely differentiates all options for the human, then even a human sampling according to LESS would exhibit behavior approaching the Boltzmann distribution. Then, based on Sec. 9.3.1, Boltzmann inference could yield superior results. However, this experiment suggest that in practical rather than adversarial cases, it is still preferable to use LESS inference on an incomplete set of features. Further, it is always possible to default in LESS to using the trajectory space directly for the similarity metric s and not rely on features.

9.4 Experiments: Robust Inference for High-DOF Arms

Sec. 9.3 teased that Boltzmann inference performance is highly dependent on the structure of the environment, and, more precisely, the feature space density induced by all possible trajectories. However, we demonstrated this on a toy task with simulated human data and ground truth access. We now put the same hypothesis to test in a real world high-dimensional scenario with a 7DoF robotic manipulator and real human demonstrations, without access to the full trajectory space nor the ground truth reward.

9.4.1 Single Demonstration Inference

Study Goal. Since for such an environment calculating the denominator in (9.3) exactly is intractable, practitioners typically *sample* the space of trajectories, obtaining varying subsets. Given the Boltzmann model's high dependency on the feature space density, we speculate that different sample sets would result in vastly varying inference results. In this section, we investigate how LESS can mitigate this effect and help inference robustness. We collect demonstrations from participants for different tasks, and run inference using different sets of trajectory for computing the normalizer.

Manipulated Variables. We used a 2-by-5 factorial design. We manipulated the *inference model* with two levels, Boltzmann and LESS, as well as the size S of the sampled trajectory sets used for inference, with five levels: 10, 30, 100, 300, and 1000. We sample 10 different trajectory sets of each size.

Other Variables. We tested our hypothesis across three household manipulation tasks where the robot learned to carry a coffee mug from a start position to a goal according to the person's preferences. In the first task, which we dub *table*, the participants were asked to move the robot arm from start to goal while maintaining the end-effector close to the table, to prevent the mug from breaking in case of a slip. In the second task, dubbed *laptop*, the participants were instructed to avoid spilling the coffee over a laptop by providing a demonstration that keeps the robot's end-effector away from the electronic device. Lastly, in the third task, dubbed *human* we asked the participants to keep the end-effector away from their body, to avoid spilling coffee on their clothes.

In all scenarios, the robot performs inference by reasoning over three features: one feature of interest (distance from the table, distance from the laptop, and distance from the human, respectively), a second feature drawn from that set, and an efficiency feature computed as the sum of squared velocities across the trajectory.

Dependent Measures. In total, for each task T, sample size S, inference method M, and user i, we obtained 10 posterior distributions $P_{M,S}^{T,i}(\hat{\theta} \mid \xi^{T,i})$ constituting a set $\mathcal{P}_{M,S}^{T,i}$. Our goal was to test how robust (or consistent) each method's inference result was across the ten different trajectory sets. We used an aggregate Kullback-Leibler divergence as a measure of how much the posterior distributions $P \in \mathcal{P}_{M,S}^{T,i}$ differ from one another:

$$KLAggregate = -\sum_{P\in\mathcal{P}_{M,S}^{T,i}} \sum_{Q\in\mathcal{P}_{M,S}^{T,i}} \sum_{\hat{\theta}\in\Theta} P(\hat{\theta} \mid \xi^{T,i}) \log\left(\frac{Q(\hat{\theta} \mid \xi^{T,i})}{P(\hat{\theta} \mid \xi^{T,i})}\right).$$

Hypothesis.

H6: Performing single inference with LESS across multiple trajectory sets results in higher robustness and, thus, a lower *KLAggregate* measure than inference with Boltzmann.

Subject Allocation. We recruited 12 users (3 female, 9 male, aged 18-30) from the campus community to physically interact with a JACO 7DOF robot arm and provide demonstrations for three tasks. Fig. 9.7 (left) illustrates the demonstrations collected for the *table* task. Before giving demonstrations, each person was allowed a period of training with the robot in gravity compensation mode to get accustomed to interacting with the robot.

Analysis. As seen in Fig. 9.7, given two different trajectory sets, inference with each method can have drastically different outcomes. With LESS (top), we see that the resulting

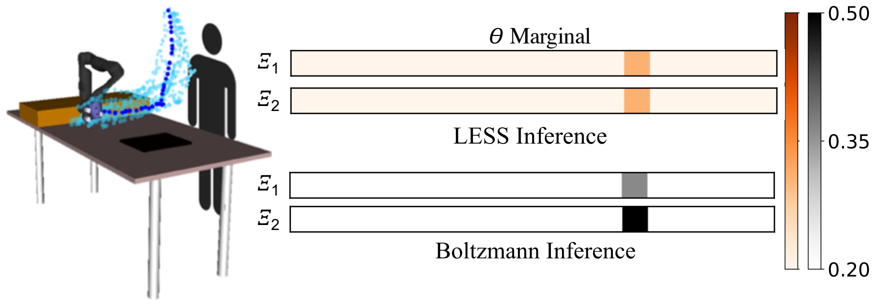


Figure 9.7: Single-demonstration (blue) inference posteriors for the *table* task with two different trajectory sets of 100 samples. The distributions reveal that both Boltzmann and LESS produce the same θ^{MAP}, but there is less variability between the LESS posteriors, leading to lower *KLAggregate*.

posterior distributions are fairly similar, whereas with Boltzmann inference (bottom), they differ in entropy/confidence.

For each sample task T, we performed a factorial repeated-measures ANOVA. The results for the *laptop* task are summarized in Fig. 9.6a. As the trend in the figure indicates, we found a significant interaction effect between inference method and sample size ($F(4, 44) = 40.37$, $p < .001$). A post-hoc Tukey HSD test revealed that LESS produced significantly lower *KLAggregate* than Boltzmann for $S = 10$, 30, and 100 ($p < 0.001$ for all), but there was no significant difference found for $S = 300$ or 1000 ($p \approx 1.00$ for both). This trend supports our hypothesis that LESS provides more robust single-demonstration inference, and it reveals that the difference in *KLAggregate* between LESS and Boltzmann disappears with increasing sample size. Results from the *table* task also support this trend, with a significant main effect of inference method.

While the *human* task did reveal a significant interaction between inference method and sample size ($F(4, 44) = 2.85$, $p < .05$) it stands apart from the other two: a post-hoc Tukey HSD test only found a difference for sample size 1000 ($p < .001$). This pattern indicates that demonstrations from this task may be generally more ambiguous and present a more difficult inference problem than the other two.

9.4.2 All Demonstrations Inference

We repeated the same experiment, except this time we run inference by aggregating all users' demonstrations for a task (batch inference). This would happen in practice if we were interested in teaching the robot about what the average user wants, rather than focusing on customizing the behavior to each user. Here, we found the opposite results, also shown in in Fig. 9.6b: LESS has higher divergence (lower robustness). We attribute this to the phenomenon described in Sec. 9.3.2. When we had only one demonstration before, Boltzmann was not robust because, depending on the set of samples, the demonstration could fall in low- *or* high-density regions, thus leading to different Boltzmann inferences for different sets. Now, with 12 demonstrations at once, the chances of one demonstration falling in a low-density area are much higher. As we've seen in Sec. 9.3.2, when there are multiple demonstrations, Boltzmann inference will be dominated by those lying in low-density areas. This leads to a more consistent posterior distribution, so long as the low-density demonstrations suggest the same reward function.

9.5 Discussion

We propose a new probabilistic human behavior model and present compelling evidence that it better captures human decision making and it attenuates inference errors that arise due to similar selections, increasing accuracy and robustness.

Our 12-person aggregate inference results in Sec. 9.4 show that LESS can lead to less robust inference. We attributed this outcome to the phenomenon in Sec. 9.3.2, but it is unclear whether this leads to less accurate inference, or whether Boltzmann is actually preferable in situations with enough varied demonstrations. Moreover, although our experiments in Sec. 9.3.3 reveal that LESS still performs better inference than Boltzmann, it is unclear whether this outcome is due to the effect of hypothesis H3 or if our method is truly unaffected by misalignment. More experiments are needed for complete clarification.

Lastly, the Mechanical Turk study in Sec. 9.2, although compelling, illustrates simplistic datasets of human choices. Further studies on human behavior in more realistic settings would be useful, but complicated by lack of access to the "ground truth" reward.

Despite these limitations, Boltzmann rationality has become so fundamental to how robots do inference and prediction, that designing a counterpart for continuous robotics domains is sorely needed. We are excited to have taken a step in this direction.

Chapter 10

Conclusion and Future Work

Robots are making their way into our society but *seamless* interaction with humans where each agent truly grasps why the other does what they do is still elusive. In this thesis, we approached interaction by explicitly tackling representation alignment, rather than hoping it will naturally emerge. However, the core of this work had the human in the teacher role feeding the robot data about their representation. In reality, seamless human-robot collaboration demands more than just passive data absorption: representation alignment should be a *bi-directional process* where the robot is an active participant in the interaction, for as the human interacts with the robot their representation of the task may also change. This opens up a broader research question: *how can robots help the human-robot team converge to a shared representation?*

I envision future work expanding upon what representation alignment should look like in human-robot interaction and bringing us closer to robots that interact with us seamlessly because we both understand why the other behaves the way they do. Us, humans, are remarkably good at this: when we coordinate and teach each other new concepts, we do so in an interactive way, where the teacher probes the learner's knowledge, the learner explains what they do and don't know, and this is all possible because we are actively trying to reach a shared representation and we have an "interface" – a common language for communicating about it. Amongst some of the challenges in translating this to robots, we need progress on making robot failures more transparent so that humans can understand what's missing in its representation, enabling the robot to explain its representation if it holds more complete task knowledge than the human, and ultimately achieving a smooth bi-directional communication. I am excited to see future work towards aligning robot and human representations, perhaps even moving beyond embodied systems: personal robots in the home or on space stations handling the human's possessions with the same care that they do, autonomous vehicles that behave according to user expectations, recommender systems that easily adjust to different preferences for frictionless personalization, drones that respect our personal space, or assistive robots that adapt to different and changing human capabilities. In this final chapter, I briefly outline a few future research directions towards such seamless human-robot interaction.

Humans Teaching Robots about their Representations

This thesis has made progress in enabling robots to learn human-aligned representations with explicit representation-specific feedback, but it opens the door to more alternative research directions in this space.

Alternative Representation Input Types. Beyond the feature traces and labels we saw in Part II, there remains a vast array of new types of human feedback about the representation to be explored: *comparisons* and *rankings* choosing or ordering behaviors more expressive of a certain feature of the representation, *equivalences* and *improvements* finding behaviors similarly or more expressive of the feature, *natural language* describing the feature, or *gaze* identifying it. We could even imagine having the robot actively select the representation query that is both most informative and most effortless, or even combining them with data augmentation techniques to provide larger amounts of feedback at once.

Representation-Specific Tasks. When the human teaches their representation explicitly, they have to break it down into the features that compose it. In Chapter 6, we saw that we can enable the robot to also *implicitly* extract the person's representation by having them solve representation-specific tasks – proxy tasks designed to learn an embedding of what matters all-at-once from their behavior. One example was a behavior similarity task, where we observed that if a person decides two robot behaviors are similar, they must also internally represent them similarly, enabling us to extract their representation. In the future, I plan to study other such proxy tasks, and even leverage unsupervised pre-training methods for improving sample efficiency. Another exciting avenue of work is designing better visual interfaces for human data collection, taking advantage of the insight that similarly represented robot behaviors should be visualized similarly.

Robots Teaching Humans about their Representations

Effective representation alignment also demands facilitating the reverse information pathway: robots that communicate new features to the human or explain their behavior by referring to their representation.

Revealing the Robot's Representation to the Human. So far the human's representation was fixed and fully captured the desired task, but the robot may sometimes hold a more complete representation: for example, it could see a useful tool that the human cannot, or it may know how to use that tool in a way that the human does not. I aim to *enable robots to share their representations* with the human and teach them new task aspects they were not aware of before. By leveraging visualization tools and natural language, I hope to make robot representations interpretable and help humans update their task understanding.

Explaining Robot Behavior. Whether the robot succeeds or fails, *explaining* its behavior can help humans have better insight into its representation. Since seeing bad behavior is often not informative of the underlying *reason* for the failure, I propose the robot can use its representation, even if incomplete, to describe what caused its behavior. Having human-aligned features should increase explainability, more clearly indicating whether the robot's representation is missing something or is poorly learned and needs repair.

Bidirectionally Evolving a Shared Representation

In some scenarios, neither the robot nor the human individually holds a complete representation, and must both communicate features they each know about. This motivates the need for developing a formalism for bidirectional communication of human and robot representations. Chapter 2 is a first step in this direction, but I want to further explore what are the best interfaces for sharing representations, from versatile and intuitive natural language to informative visualizations. By alternating between the direct (robot learning about the human's representation) and the reverse (robot teaching the human about its representation) channels of communication, I hope to enable seamlessly reaching a mutual task representation. I am particularly interested in demonstrating what benefits this could have in areas beyond learning and predicting human intended tasks, to robot policy learning, exploration, or even human-robot coordination without explicit learning. Lastly, I'm interested in studying interesting connections to game-theory, which may rigorously couple the influence between the human, the robot, and their representations.

Bibliography

[1] Pieter Abbeel, Adam Coates, and Andrew Y Ng. "Autonomous helicopter aerobatics through apprenticeship learning". In: *The International Journal of Robotics Research* 29.13 (2010), pp. 1608–1639.

[2] Pieter Abbeel and Andrew Y Ng. "Apprenticeship learning via inverse reinforcement learning". In: *Machine Learning (ICML), International Conference on*. ACM. 2004.

[3] David Abel et al. "On the expressivity of markov reward". In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 7799–7812.

[4] David Abel et al. "State abstractions for lifelong reinforcement learning". In: *International Conference on Machine Learning*. PMLR. 2018, pp. 10–19.

[5] F. Abi-Farraj, C. Pacchierotti, and P. R. Giordano. "User Evaluation of a Haptic-Enabled Shared-Control Approach for Robotic Telemanipulation". In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2018, pp. 1–9. DOI: 10.1109/IROS.2018.8594030.

[6] Sameer Agarwal et al. "Generalized non-metric multidimensional scaling". In: *Artificial Intelligence and Statistics*. PMLR. 2007, pp. 11–18.

[7] N. Aghasadeghi and T. Bretl. "Maximum entropy inverse reinforcement learning in continuous state spaces with path integrals". In: *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2011, pp. 1561–1566. DOI: 10.1109/IROS.2011.6094679.

[8] Pulkit Agrawal. "The Task Specification Problem". In: *Conference on Robot Learning*. PMLR. 2022, pp. 1745–1751.

[9] Peter Aigner and Brenan McCarragher. "Human integration into robot control utilising potential fields". In: *Proceedings of International Conference on Robotics and Automation*. Vol. 1. IEEE. 1997, pp. 291–296.

[10] Guillaume Alain and Yoshua Bengio. "Understanding intermediate layers using linear classifier probes". In: *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Workshop Track Proceedings*. OpenReview.net, 2017. URL: https://openreview.net/forum?id=HJ4-rAVtl.

[11] Alnour Alharin, Thanh-Nam Doan, and Mina Sartipi. "Reinforcement learning interpretation methods: A survey". In: *IEEE Access* 8 (2020), pp. 171058–171077.

[12] Ehsan Amid, Aristides Gionis, and Antti Ukkonen. "A Kernel-Learning Approach to Semi-supervised Clustering with Relative Distance Comparisons". In: vol. 9284. Sept. 2015. ISBN: 978-3-319-23527-1. DOI: 10.1007/978-3-319-23528-8_14.

[13] Dario Amodei and Jack Clark. "Faulty Reward Functions in the Wild". In: (2016). URL: https://blog.openai.%20com/faulty-reward-functions/.

[14] Dario Amodei et al. *Concrete Problems in AI Safety*. 2016. arXiv: 1606.06565 [cs.AI].

[15] Ankesh Anand et al. "Unsupervised State Representation Learning in Atari". In: *Advances in Neural Information Processing Systems*. Ed. by H. Wallach et al. Vol. 32. Curran Associates, Inc., 2019. URL: https://proceedings.neurips.cc/paper/2019/file/6fb52e71b837628ac16539c1ff911667-Paper.pdf.

[16] Joel AE Andersson et al. "CasADi: a software framework for nonlinear optimization and optimal control". In: *Mathematical Programming Computation* (2019).

[17] Brenna D Argall et al. "A survey of robot learning from demonstration". In: *Robotics and autonomous systems* 57.5 (2009), pp. 469–483.

[18] Brenna D. Argall, Eric L. Sauser, and Aude G. Billard. "Tactile Guidance for Policy Adaptation". In: *Found. Trends Robot* 1.2 (Feb. 2011), pp. 79–133. ISSN: 1935-8253. DOI: 10.1561/2300000012. URL: http://dx.doi.org/10.1561/2300000012.

[19] Martín Arjovsky et al. "Invariant Risk Minimization". In: *ArXiv* abs/1907.02893 (2019).

[20] Sanjeev Arora et al. "Provable Representation Learning for Imitation Learning via Bi-Level Optimization". In: *Proceedings of the 37th International Conference on Machine Learning*. ICML'20. JMLR.org, 2020.

[21] Tsz-Chiu Au et al. "SHOP2: An HTN Planning System". In: *CoRR* abs/1106.4869 (2011). arXiv: 1106.4869. URL: http://arxiv.org/abs/1106.4869.

[22] Yusuf Aytar et al. "Playing Hard Exploration Games by Watching YouTube". In: *Proceedings of the 32nd International Conference on Neural Information Processing Systems*. NIPS'18. Montréal, Canada: Curran Associates Inc., 2018, pp. 2935–2945.

[23] Monica Babes et al. "Apprenticeship learning about multiple intentions". In: *ICML*. 2011.

[24] Andrea Bajcsy et al. "Learning from Physical Human Corrections, One Feature at a Time". In: *Proceedings of the 2018 ACM/IEEE International Conference on Human-Robot Interaction*. HRI '18. Chicago, IL, USA: ACM, 2018, pp. 141–149. ISBN: 978-1-4503-4953-6. DOI: 10.1145/3171221.3171267. URL: http://doi.acm.org/10.1145/3171221.3171267.

[25] Andrea Bajcsy et al. "Learning Robot Objectives from Physical Human Interaction". In: *Proceedings of the 1st Annual Conference on Robot Learning*. Ed. by Sergey Levine, Vincent Vanhoucke, and Ken Goldberg. Vol. 78. Proceedings of Machine Learning Research. PMLR, 2017, pp. 217–226. URL: http://proceedings.mlr.press/v78/bajcsy17a.html.

[26] Bowen Baker et al. "Video PreTraining (VPT): Learning to Act by Watching Unlabeled Online Videos". In: *CoRR* abs/2206.11795 (2022). DOI: 10.48550/arXiv.2206.11795. arXiv: 2206.11795. URL: https://doi.org/10.48550/arXiv.2206.11795.

[27] Chris L Baker, Joshua B Tenenbaum, and Rebecca R Saxe. "Goal inference as inverse planning". In: *Proceedings of the Annual Meeting of the Cognitive Science Society*. Vol. 29. 29. 2007.

[28] Vassileios Balntas et al. "Learning local feature descriptors with triplets and shallow convolutional neural networks". In: Jan. 2016, pp. 119.1–119.11. DOI: 10.5244/C.30.119.

[29] Peter Bartlett, Vitaly Maiorov, and Ron Meir. "Almost Linear VC Dimension Bounds for Piecewise Polynomial Networks". In: *Advances in Neural Information Processing Systems*. Ed. by M. Kearns, S. Solla, and D. Cohn. Vol. 11. MIT Press, 1998.

[30] Peter L. Bartlett, Dylan J. Foster, and Matus Telgarsky. "Spectrally-normalized margin bounds for neural networks". In: *NIPS*. 2017.

[31] Eric Baum and David Haussler. "What Size Net Gives Valid Generalization?" In: *Advances in Neural Information Processing Systems*. Ed. by D. Touretzky. Vol. 1. Morgan-Kaufmann, 1988. URL: https://proceedings.neurips.cc/paper/1988/file/1d7f7abc18fcb43975065399b0d1e48e-Paper.pdf.

[32] Moshe Ben-Akiva. "Structure of Passenger Travel Demand Models". In: *Transportation Research Record* 526 (Aug. 1973).

[33] Tom Bewley and Jonathan Lawry. "Tripletree: A versatile interpretable representation of black box agents and their environments". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 35. 2021, pp. 11415–11422.

[34] Lorenz T Biegler and Victor M Zavala. "Large-scale nonlinear programming using IPOPT: An integrating framework for enterprise-wide dynamic optimization". In: *Computers & Chemical Engineering* (2009).

[35] Daniel Birman and Justin L. Gardner. "A flexible readout mechanism of human sensory representations". In: *Nature Communications* 10.1 (2019), p. 3500. DOI: 10.1038/s41467-019-11448-7. URL: https://doi.org/10.1038/s41467-019-11448-7.

[36] Erdem Biyik and Dorsa Sadigh. "Batch active preference-based learning of reward functions". In: *Conference on robot learning*. PMLR. 2018, pp. 519–528.

[37] Erdem Biyik et al. “Batch Active Learning Using Determinantal Point Processes”. In: *CoRR* abs/1906.07975 (2019). arXiv: 1906.07975.

[38] A. Bobu et al. “Quantifying Hypothesis Space Misspecification in Learning From Human–Robot Demonstrations and Physical Corrections”. In: *IEEE Transactions on Robotics* (2020), pp. 1–20.

[39] Andreea Bobu et al. *Aligning Robot and Human Representations*. 2023. arXiv: 2302.01928 [cs.RO].

[40] Andreea Bobu et al. “Feature Expansive Reward Learning: Rethinking Human Input”. In: *Proceedings of the 2021 ACM/IEEE International Conference on Human-Robot Interaction*. HRI ’21. Boulder, CO, USA: Association for Computing Machinery, 2021, pp. 216–224. ISBN: 9781450382892. DOI: 10.1145/3434073.3444667. URL: https://doi.org/10.1145/3434073.3444667.

[41] Andreea Bobu et al. “Inducing Structure in Reward Learning by Learning Features”. In: *The International Journal of Robotics Research* (2022). DOI: 10.1177/02783649221078031. URL: https://doi.org/10.1177/02783649221078031.

[42] Andreea Bobu et al. “Learning Perceptual Concepts by Bootstrapping From Human Queries”. In: *IEEE Robotics Autom. Lett.* 7.4 (2022), pp. 11260–11267. DOI: 10.1109/LRA.2022.3196164. URL: https://doi.org/10.1109/LRA.2022.3196164.

[43] Andreea Bobu et al. “Learning under Misspecified Objective Spaces”. In: *Proceedings of The 2nd Conference on Robot Learning*. Ed. by Aude Billard et al. Vol. 87. Proceedings of Machine Learning Research. PMLR, 2018, pp. 796–805. URL: http://proceedings.mlr.press/v87/bobu18a.html.

[44] Andreea Bobu et al. “LESS is More: Rethinking Probabilistic Models of Human Behavior”. In: *Proceedings of the 2020 ACM/IEEE International Conference on Human-Robot Interaction*. New York, NY, USA: Association for Computing Machinery, 2020, pp. 429–437. ISBN: 9781450367462. URL: https://doi.org/10.1145/3319502.3374811.

[45] Andreea Bobu et al. “SIRL: Similarity-Based Implicit Representation Learning”. In: *Proceedings of the 2023 ACM/IEEE International Conference on Human-Robot Interaction*. HRI ’23. Stockholm, Sweden: Association for Computing Machinery, 2023, pp. 565–574. ISBN: 9781450399647. DOI: 10.1145/3568162.3576989. URL: https://doi.org/10.1145/3568162.3576989.

[46] Tyler Bonnen, Daniel L.K. Yamins, and Anthony D. Wagner. “When the ventral visual stream is not enough: A deep learning account of medial temporal lobe involvement in perception”. In: *Neuron* 109.17 (2021), 2755–2766.e6. ISSN: 0896-6273. DOI: https://doi.org/10.1016/j.neuron.2021.06.018. URL: https://www.sciencedirect.com/science/article/pii/S0896627321004591.

[47] Ralph Allan Bradley and Milton E Terry. "Rank analysis of incomplete block designs: I. The method of paired comparisons". In: *Biometrika* 39.3/4 (1952), pp. 324–345.

[48] Darius Braziunas and Craig Boutilier. "Elicitation of Factored Utilities". In: *AI Magazine* 29.4 (Dec. 2008), p. 79. DOI: 10.1609/aimag.v29i4.2203. URL: https://ojs.aaai.org/index.php/aimagazine/article/view/2203.

[49] Daniel Brown et al. "Extrapolating beyond suboptimal demonstrations via inverse reinforcement learning from observations". In: *International Conference on Machine Learning*. PMLR. 2019, pp. 783–792.

[50] Daniel Brown et al. "Safe Imitation Learning via Fast Bayesian Reward Inference from Preferences". In: *Proceedings of the 37th International Conference on Machine Learning*. Ed. by Hal Daumé III and Aarti Singh. Vol. 119. Proceedings of Machine Learning Research. PMLR, 2020, pp. 1165–1177. URL: http://proceedings.mlr.press/v119/brown20a.html.

[51] Daniel S Brown, Yuchen Cui, and Scott Niekum. "Risk-aware active inverse reinforcement learning". In: *Conference on Robot Learning*. PMLR. 2018, pp. 362–372.

[52] Daniel S Brown, Shin-Young Jung, and Michael A Goodrich. "Balancing human and inter-agent influences for shared control of bio-inspired collectives". In: *2014 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. IEEE. 2014, pp. 4123–4128.

[53] Tom Brown et al. "Language models are few-shot learners". In: *Advances in neural information processing systems* 33 (2020), pp. 1877–1901.

[54] Sven Buechel and Udo Hahn. "EmoBank: Studying the Impact of Annotation Perspective and Representation Format on Dimensional Emotion Analysis". In: *EACL*. 2017.

[55] Kalesha Bullard, Sonia Chernova, and Andrea Lockerd Thomaz. "Human-Driven Feature Selection for a Robotic Agent Learning Classification Tasks from Demonstration". In: *2018 IEEE International Conference on Robotics and Automation, ICRA 2018, Brisbane, Australia, May 21-25, 2018*. IEEE, 2018, pp. 6923–6930. DOI: 10.1109/ICRA.2018.8461012. URL: https://doi.org/10.1109/ICRA.2018.8461012.

[56] Suci G. C.E. Osgood. *The measurement of meaning*. University of Illinois Press, 1957.

[57] Maya Cakmak and Andrea L. Thomaz. "Designing robot learners that ask good questions". In: *2012 7th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*. 2012, pp. 17–24. DOI: 10.1145/2157689.2157693.

[58] Maya Cakmak and Andrea Lockerd Thomaz. "Designing robot learners that ask good questions". In: *International Conference on Human-Robot Interaction, HRI'12, Boston, MA, USA - March 05 - 08, 2012*. Ed. by Holly A. Yanco et al. ACM, 2012, pp. 17–24. DOI: 10.1145/2157689.2157693. URL: https://doi.org/10.1145/2157689.2157693.

[59] Frederick Callaway, Antonio Rangel, and Thomas L Griffiths. “Fixation patterns in simple choice reflect optimal information sampling”. In: *PLoS computational biology* 17.3 (2021), e1008863.

[60] Carlos Celemin, Javier Ruiz-del-Solar, and Jens Kober. “A fast hybrid reinforcement learning framework with human corrective feedback”. In: *Autonomous Robots* 43.5 (2019), pp. 1173–1186. ISSN: 1573-7527. DOI: 10.1007/s10514-018-9786-6. URL: https://doi.org/10.1007/s10514-018-9786-6.

[61] Angel X. Chang et al. *ShapeNet: An Information-Rich 3D Model Repository*. Tech. rep. arXiv:1512.03012 [cs.GR]. Stanford University — Princeton University — Toyota Technological Institute at Chicago, 2015.

[62] Annie S. Chen, Suraj Nair, and Chelsea Finn. “Learning Generalizable Robotic Reward Functions from "In-The-Wild" Human Videos”. In: *Robotics: Science and Systems XVII, Virtual Event, July 12-16, 2021*. Ed. by Dylan A. Shell, Marc Toussaint, and M. Ani Hsieh. 2021. DOI: 10.15607/RSS.2021.XVII.012. URL: https://doi.org/10.15607/RSS.2021.XVII.012.

[63] Kevin Chen et al. “Learning Hierarchical Task Networks with Preferences from Unannotated Demonstrations”. In: *4th Conference on Robot Learning, CoRL 2020, 16-18 November 2020, Virtual Event / Cambridge, MA, USA*. Ed. by Jens Kober, Fabio Ramos, and Claire J. Tomlin. Vol. 155. Proceedings of Machine Learning Research. PMLR, 2020, pp. 1572–1581. URL: https://proceedings.mlr.press/v155/chen21d.html.

[64] Ricky T. Q. Chen et al. “Isolating Sources of Disentanglement in Variational Autoencoders”. In: *Advances in Neural Information Processing Systems*. Ed. by S. Bengio et al. Vol. 31. Curran Associates, Inc., 2018.

[65] Ting Chen et al. “A simple framework for contrastive learning of visual representations”. In: *International conference on machine learning*. PMLR. 2020, pp. 1597–1607.

[66] Xi Chen et al. “InfoGAN: Interpretable Representation Learning by Information Maximizing Generative Adversarial Nets”. In: *Proceedings of the 30th International Conference on Neural Information Processing Systems*. NIPS’16. Barcelona, Spain: Curran Associates Inc., 2016, pp. 2180–2188. ISBN: 9781510838819.

[67] Xin Chen et al. “An Empirical Investigation of Representation Learning for Imitation”. In: *CoRR* abs/2205.07886 (2022). DOI: 10.48550/arXiv.2205.07886. arXiv: 2205.07886. URL: https://doi.org/10.48550/arXiv.2205.07886.

[68] Jaedeug Choi and Kee-Eung Kim. “Bayesian nonparametric feature construction for inverse reinforcement learning”. In: *Twenty-Third International Joint Conference on Artificial Intelligence*. 2013.

[69] Jaedeug Choi and Kee-Eung Kim. "Inverse reinforcement learning in partially observable environments". In: *Journal of Machine Learning Research* 12.Mar (2011), pp. 691–730.

[70] Jaedeug Choi and Kee-Eung Kim. "Nonparametric Bayesian inverse reinforcement learning for multiple reward functions". In: *Advances in Neural Information Processing Systems* 25 (2012).

[71] Paul F Christiano et al. "Deep Reinforcement Learning from Human Preferences". In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon et al. Vol. 30. Curran Associates, Inc., 2017.

[72] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. "Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs)". In: *arXiv: Learning* (2016).

[73] Adam Coates and A. Ng. "Learning Feature Representations with K-Means". In: *Neural Networks: Tricks of the Trade*. 2012.

[74] Erwin Coumans and Yunfei Bai. *PyBullet, a Python module for physics simulation for games, robotics and machine learning*. http://pybullet.org. 2016.

[75] Jacob W Crandall et al. "Human-swarm interaction as shared control: Achieving flexible fault-tolerant systems". In: *International Conference on Engineering Psychology and Cognitive Ergonomics*. Springer. 2017, pp. 266–284.

[76] Angel Andres Daruna, Devleena Das, and Sonia Chernova. "Explainable Knowledge Graph Embedding: Inference Reconciliation for Knowledge Inferences Supporting Robot Actions". In: *CoRR* abs/2205.01836 (2022). DOI: 10.48550/arXiv.2205.01836. arXiv: 2205.01836. URL: https://doi.org/10.48550/arXiv.2205.01836.

[77] Angel Andres Daruna et al. "Towards Robust One-shot Task Execution using Knowledge Graph Embeddings". In: *IEEE International Conference on Robotics and Automation, ICRA 2021, Xi'an, China, May 30 - June 5, 2021*. IEEE, 2021, pp. 11118–11124. DOI: 10.1109/ICRA48506.2021.9561782. URL: https://doi.org/10.1109/ICRA48506.2021.9561782.

[78] Pieter-Tjerk De Boer et al. "A tutorial on the cross-entropy method". In: *Annals of operations research* 134.1 (2005), pp. 19–67.

[79] Gerard Debreu. In: *The American Economic Review* 50.1 (1960), pp. 186–188. ISSN: 00028282. URL: http://www.jstor.org/stable/1813477.

[80] Tristan Deleu et al. "Bayesian Structure Learning with Generative Flow Networks". In: *CoRR* abs/2202.13903 (2022). arXiv: 2202.13903. URL: https://arxiv.org/abs/2202.13903.

[81] Frank Dellaert. "Expectation-Maximization Algorithm". In: *Encyclopedia of Machine Learning*. 2010.

[82] Çağatay Demiralp, Michael S Bernstein, and Jeffrey Heer. "Learning perceptual kernels for visualization design". In: *IEEE transactions on visualization and computer graphics* 20.12 (2014), pp. 1933–1942.

[83] Xinke Deng et al. "PoseRBPF: A Rao–Blackwellized Particle Filter for 6-D Object Pose Tracking". In: *IEEE Transactions on Robotics* 37.5 (2021), pp. 1328–1342. DOI: 10.1109/TRO.2021.3056043.

[84] Ruta Desai et al. "Geppetto: Enabling Semantic Design of Expressive Robot Behaviors". In: *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems* (2019).

[85] Jacob Devlin et al. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: *ArXiv* abs/1810.04805 (2019).

[86] Christos Dimitrakakis and Constantin A Rothkopf. "Bayesian multitask inverse reinforcement learning". In: *European workshop on reinforcement learning*. Springer. 2011, pp. 273–284.

[87] Carl Doersch, Abhinav Kumar Gupta, and Alexei A. Efros. "Unsupervised Visual Representation Learning by Context Prediction". In: *2015 IEEE International Conference on Computer Vision (ICCV)* (2015), pp. 1422–1430.

[88] A. D. Dragan et al. "Movement primitives via optimization". In: *2015 IEEE International Conference on Robotics and Automation (ICRA)*. May 2015, pp. 2339–2346. DOI: 10.1109/ICRA.2015.7139510.

[89] Anca D Dragan and Siddhartha S Srinivasa. "A policy-blending formalism for shared control". In: *The International Journal of Robotics Research* 32.7 (2013), pp. 790–805. DOI: 10.1177/0278364913490324. eprint: https://doi.org/10.1177/0278364913490324. URL: https://doi.org/10.1177/0278364913490324.

[90] Anca D. Dragan, Kenton C.T. Lee, and Siddhartha S. Srinivasa. "Legibility and predictability of robot motion". In: *2013 8th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*. 2013, pp. 301–308. DOI: 10.1109/HRI.2013.6483603.

[91] Simon S. Du et al. *Few-Shot Learning via Learning the Representation, Provably*. 2020. DOI: 10.48550/ARXIV.2002.09434. URL: https://arxiv.org/abs/2002.09434.

[92] Ahmetcan Erdogan and Brenna D Argall. "The effect of robotic wheelchair control paradigm and interface on user performance, effort and preference: an experimental assessment". In: *Robotics and Autonomous Systems* 94 (2017), pp. 282–297.

[93] Linus Ericsson, Henry Gouk, and Timothy M. Hospedales. "How Well Do Self-Supervised Models Transfer?" In: *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2021, virtual, June 19-25, 2021*. Computer Vision Foundation / IEEE, 2021, pp. 5414–5423. DOI: 10.1109/CVPR46437.2021.00537. URL: https://openaccess.thecvf.com/content/CVPR2021/html/Ericsson%5C_How%5C_Well%5C_Do%5C_Self-Supervised%5C_Models%5C_Transfer%5C_CVPR%5C_2021%5C_paper.html.

[94] Benjamin Eysenbach et al. "Diversity is all you need: Learning skills without a reward function". In: *arXiv preprint arXiv:1802.06070* (2018).

[95] Chelsea Finn, Pieter Abbeel, and Sergey Levine. "Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks". In: *Proceedings of the 34th International Conference on Machine Learning - Volume 70*. ICML'17. Sydney, NSW, Australia: JMLR.org, 2017, pp. 1126–1135.

[96] Chelsea Finn, Sergey Levine, and Pieter Abbeel. "Guided Cost Learning: Deep Inverse Optimal Control via Policy Optimization". In: *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*. ICML'16. New York, NY, USA: JMLR.org, 2016, pp. 49–58.

[97] Chelsea Finn et al. "Learning Visual Feature Spaces for Robotic Manipulation with Deep Spatial Autoencoders". In: *CoRR* abs/1509.06113 (2015). arXiv: 1509.06113. URL: http://arxiv.org/abs/1509.06113.

[98] Jaime F Fisac et al. "Probabilistically Safe Robot Planning with Confidence-Based Human Predictions". In: *Robotics: Science and Systems (RSS)* (2018).

[99] Jaime F. Fisac et al. "Pragmatic-Pedagogic Value Alignment". In: *Robotics Research, The 18th International Symposium, ISRR 2017, Puerto Varas, Chile, December 11-14, 2017*. Ed. by Nancy M. Amato et al. Vol. 10. Springer Proceedings in Advanced Robotics. Springer, 2017, pp. 49–57. DOI: 10.1007/978-3-030-28619-4_7. URL: https://doi.org/10.1007/978-3-030-28619-4%5C_7.

[100] David Fridovich-Keil et al. "Confidence-aware motion prediction for real-time collision avoidance". In: *International Journal of Robotics Research* (2019).

[101] Jie Fu and Ufuk Topcu. "Pareto efficiency in synthesizing shared autonomy policies with temporal logic constraints". In: *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2015, pp. 361–368.

[102] Justin Fu, Katie Luo, and Sergey Levine. "Learning Robust Rewards with Adverserial Inverse Reinforcement Learning". In: *International Conference on Learning Representations*. 2018. URL: https://openreview.net/forum?id=rkHywl-A-.

[103] Justin Fu et al. "Variational Inverse Control with Events: A General Framework for Data-Driven Reward Definition". In: *arXiv preprint* arXiv:1805.11686 (2018).

[104] Javier Garcıa and Fernando Fernández. "A comprehensive survey on safe reinforcement learning". In: *Journal of Machine Learning Research* 16.1 (2015), pp. 1437–1480.

[105] Dominic Gates. "Flawed analysis, failed oversight: How Boeing, FAA certified the suspect 737 MAX flight control system". In: *The Seattle Times* (Mar. 2019). URL: https://www.seattletimes.com/business/boeing-aerospace/failed-certification-faa-missed-safety-issues-in-the-737-max-system-implicated-in-the-lion-air-crash/.

[106] Dibya Ghosh, Abhishek Gupta, and Sergey Levine. "Learning Actionable Representations with Goal Conditioned Policies". In: *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. URL: https://openreview.net/forum?id=Hye9lnCct7.

[107] Michael J. Gielniak and Andrea L. Thomaz. "Generating anticipation in robot motion". In: *2011 RO-MAN*. 2011, pp. 449–454. DOI: 10.1109/ROMAN.2011.6005255.

[108] Claire Glanois et al. "A Survey on Interpretable Reinforcement Learning". In: *arXiv preprint arXiv:2112.13112* (2021).

[109] Adam Gleave and Oliver Habryka. "Multi-task maximum entropy inverse reinforcement learning". In: *arXiv preprint arXiv:1805.08882* (2018).

[110] Ray C Goertz. "Manipulators used for handling radioactive materials". In: *Human factors in technology* (1963), pp. 425–443.

[111] Noah Golowich, Alexander Rakhlin, and Ohad Shamir. "Size-Independent Sample Complexity of Neural Networks". In: *Proceedings of the 31st Conference On Learning Theory*. Ed. by Sébastien Bubeck, Vianney Perchet, and Philippe Rigollet. Vol. 75. Proceedings of Machine Learning Research. PMLR, 2018, pp. 297–299. URL: https://proceedings.mlr.press/v75/golowich18a.html.

[112] Ankit Goyal et al. *IFOR: Iterative Flow Minimization for Robotic Object Rearrangement*. 2022. arXiv: 2202.00732 [cs.RO].

[113] Samuel Greydanus et al. "Visualizing and understanding atari agents". In: *International conference on machine learning*. PMLR. 2018, pp. 1792–1801.

[114] Faruk Gul, Paulo Natenzon, and Wolfgang Pesendorfer. "Random Choice as Behavioral Optimization". In: 2014.

[115] Martin Günther et al. "Context-aware 3D object anchoring for mobile robots". In: *Robotics Auton. Syst.* 110 (2018), pp. 12–32. DOI: 10.1016/j.robot.2018.08.016. URL: https://doi.org/10.1016/j.robot.2018.08.016.

[116] Piyush Gupta et al. "Explain your move: Understanding agent actions using focused feature saliency". In: *arXiv preprint arXiv:1912.12191* (2019).

[117] Todd M. Gureckis et al. "psiTurk: An open-source framework for conducting replicable behavioral experiments online". In: *Behavior Research Methods* 48.3 (2016), pp. 829–842. ISSN: 1554-3528. DOI: 10.3758/s13428-015-0642-8. URL: https://doi.org/10.3758/s13428-015-0642-8.

[118] Reymundo Gutierrez et al. "Incremental Task Modification via Corrective Demonstrations". In: *2018 IEEE International Conference on Robotics and Automation (ICRA)* (2018), pp. 1126–1133.

[119] David Ha and Jürgen Schmidhuber. "Recurrent World Models Facilitate Policy Evolution". In: *Advances in Neural Information Processing Systems*. Ed. by S. Bengio et al. Vol. 31. Curran Associates, Inc., 2018. URL: https://proceedings.neurips.cc/paper/2018/file/2de5d16682c3c35007e4e92982f1a2ba-Paper.pdf.

[120] Pim de Haan, Dinesh Jayaraman, and Sergey Levine. "Causal Confusion in Imitation Learning". In: *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*. Ed. by Hanna M. Wallach et al. 2019, pp. 11693–11704. URL: https://proceedings.neurips.cc/paper/2019/hash/947018640bf36a2bb609d3557a285329-Abstract.html.

[121] Pim de Haan, Dinesh Jayaraman, and Sergey Levine. "Causal Confusion in Imitation Learning". In: *Advances in Neural Information Processing Systems*. Ed. by H. Wallach et al. Vol. 32. Curran Associates, Inc., 2019.

[122] Dylan Hadfield-Menell et al. "Cooperative Inverse Reinforcement Learning". In: *Proceedings of the 30th International Conference on Neural Information Processing Systems*. NIPS'16. Barcelona, Spain: Curran Associates Inc., 2016, pp. 3916–3924. ISBN: 978-1-5108-3881-9. URL: http://dl.acm.org/citation.cfm?id=3157382.3157535.

[123] Dylan Hadfield-Menell et al. "Inverse Reward Design". In: *NIPS*. 2017.

[124] Dylan Hadfield-Menell et al. "Inverse reward design". In: *Advances in neural information processing systems* 30 (2017).

[125] Danijar Hafner et al. "Dream to Control: Learning Behaviors by Latent Imagination". In: *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. URL: https://openreview.net/forum?id=S1lOTC4tDS.

[126] Sandra G. Hart and Lowell E. Staveland. "Development of NASA-TLX (Task Load Index): Results of Empirical and Theoretical Research". In: *Human Mental Workload*. Ed. by Peter A. Hancock and Najmedin Meshkati. Vol. 52. Advances in Psychology. North-Holland, 1988, pp. 139–183. DOI: https://doi.org/10.1016/S0166-4115(08)62386-9. URL: http://www.sciencedirect.com/science/article/pii/S0166411508623869.

[127] Nick Harvey, Christopher Liaw, and Abbas Mehrabian. "Nearly-tight VC-dimension bounds for piecewise linear neural networks". In: *Proceedings of the 2017 Conference on Learning Theory*. Ed. by Satyen Kale and Ohad Shamir. Vol. 65. Proceedings of Machine Learning Research. PMLR, 2017, pp. 1064–1068. URL: https://proceedings.mlr.press/v65/harvey17a.html.

[128] Luis Haug, Sebastian Tschiatschek, and Adish Singla. "Teaching inverse reinforcement learners via features and demonstrations". In: *Advances in Neural Information Processing Systems*. 2018, pp. 8464–8473.

[129] Bradley Hayes and Brian Scassellati. "Discovering task constraints through observation and active learning". In: *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems, Chicago, IL, USA, September 14-18, 2014*. IEEE, 2014, pp. 4442–4449. DOI: 10.1109/IROS.2014.6943191. URL: https://doi.org/10.1109/IROS.2014.6943191.

[130] Bradley Hayes and Julie A Shah. "Improving robot controller transparency through autonomous policy explanation". In: *2017 12th ACM/IEEE International Conference on Human-Robot Interaction (HRI*. IEEE. 2017, pp. 303–312.

[131] P. Henry et al. "Learning to navigate through crowded environments". In: *2010 IEEE International Conference on Robotics and Automation*. 2010, pp. 981–986. DOI: 10.1109/ROBOT.2010.5509772.

[132] Ayonga Hereid et al. "Rapid trajectory optimization using c-frost with illustration on a cassie-series dynamic walking biped". In: *International Conference on Intelligent Robots and Systems (IROS)*. 2019, pp. 4722–4729.

[133] Irina Higgins et al. "beta-VAE: Learning Basic Visual Concepts with a Constrained Variational Framework". In: *ICLR*. 2017.

[134] Irina Higgins et al. "DARLA: Improving Zero-Shot Transfer in Reinforcement Learning". In: *ICML*. 2017.

[135] Sophie Hilgard et al. "Learning Representations by Humans, for Humans". In: *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*. Ed. by Marina Meila and Tong Zhang. Vol. 139. Proceedings of Machine Learning Research. PMLR, 2021, pp. 4227–4238. URL: http://proceedings.mlr.press/v139/hilgard21a.html.

[136] Jonathan Ho and Stefano Ermon. "Generative Adversarial Imitation Learning". In: *Advances in Neural Information Processing Systems 29*. Ed. by D. D. Lee et al. Curran Associates, Inc., 2016, pp. 4565–4573. URL: http://papers.nips.cc/paper/6391-generative-adversarial-imitation-learning.pdf.

[137] Mark K Ho. "The value of abstraction". In: *Current opinion in behavioral sciences* 29 (2019).

[138] Neville Hogan. "Impedance Control: An Approach to Manipulation". In: *Journal of Dynamic Systems, Measurement, and Control* 107.1 (1985), pp. 8–16. ISSN: 0022-0434. DOI: 10.1115/1.3140713. URL: http://dx.doi.org/10.1115/1.3140713.

[139] Daniel Holden, Jun Saito, and Taku Komura. "A Deep Learning Framework for Character Motion Synthesis and Editing". In: *ACM Trans. Graph.* 35.4 (July 2016). ISSN: 0730-0301. DOI: 10.1145/2897824.2925975.

[140] Yordan Hristov et al. “Disentangled Relational Representations for Explaining and Learning from Demonstration”. In: *3rd Annual Conference on Robot Learning, CoRL 2019, Osaka, Japan, October 30 - November 1, 2019, Proceedings*. Ed. by Leslie Pack Kaelbling, Danica Kragic, and Komei Sugiura. Vol. 100. Proceedings of Machine Learning Research. PMLR, 2019, pp. 870–884.

[141] Chao Huang, Wenhao Luo, and Rui Liu. “Meta Preference Learning for Fast User Adaptation in Human-Supervisory Multi-Robot Deployments”. In: *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2021, pp. 5851–5856.

[142] Borja Ibarz et al. “Reward learning from human preferences and demonstrations in Atari”. In: *Advances in Neural Information Processing Systems*. Ed. by S. Bengio et al. Vol. 31. Curran Associates, Inc., 2018, pp. 8011–8023. URL: https://proceedings.neurips.cc/paper/2018/file/8cbe9ce23f42628c98f80fa0fac8b19a-Paper.pdf.

[143] Andrew Ilyas et al. “Adversarial examples are not bugs, they are features”. In: *Advances in neural information processing systems* 32 (2019).

[144] Ashesh Jain et al. “Learning preferences for manipulation tasks from online coactive feedback”. In: *The International Journal of Robotics Research* 34.10 (2015), pp. 1296–1313.

[145] Shervin Javdani et al. “Shared autonomy via hindsight optimization for teleoperation and teaming”. In: *The International Journal of Robotics Research* 37.7 (2018), pp. 717–742. DOI: 10.1177/0278364918776060. eprint: https://doi.org/10.1177/0278364918776060. URL: https://doi.org/10.1177/0278364918776060.

[146] E. T. Jaynes. “Information Theory and Statistical Mechanics”. In: vol. 106. American Physical Society, May 1957, pp. 620–630. DOI: 10.1103/PhysRev.106.620. URL: https://link.aps.org/doi/10.1103/PhysRev.106.620.

[147] Hong J Jeon and Anca D Dragan. “Configuration Space Metrics”. In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2018, pp. 5101–5108.

[148] 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* 33 (2020), pp. 4415–4426.

[149] Leslie Pack Kaelbling, Michael L. Littman, and Anthony R. Cassandra. “Planning and acting in partially observable stochastic domains”. In: *Artificial Intelligence* 101.1 (1998), pp. 99–134. ISSN: 0004-3702. DOI: https://doi.org/10.1016/S0004-3702(98)00023-X. URL: http://www.sciencedirect.com/science/article/pii/S000437029800023X.

[150] M. Kalakrishnan et al. “Learning objective functions for manipulation”. In: *2013 IEEE International Conference on Robotics and Automation*. May 2013, pp. 1331–1336. DOI: 10.1109/ICRA.2013.6630743.

[151] R E Kalman. “When Is a Linear Control System Optimal?” In: *Journal of Basic Engineering* 86.1 (Mar. 1964), pp. 51–60. ISSN: 0098-2202. URL: http://dx.doi.org/10.1115/1.3653115.

[152] Martin Karlsson, Anders Robertsson, and Rolf Johansson. “Autonomous interpretation of demonstrations for modification of dynamical movement primitives”. In: *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2017, pp. 316–321.

[153] Marek Karpinski and Angus Macintyre. “Polynomial Bounds for VC Dimension of Sigmoidal and General Pfaffian Neural Networks”. In: *Journal of Computer and System Sciences* 54.1 (1997), pp. 169–176. ISSN: 0022-0000. DOI: https://doi.org/10.1006/jcss.1997.1477. URL: https://www.sciencedirect.com/science/article/pii/S002200009791477X.

[154] Kei Kase et al. “Transferable Task Execution from Pixels through Deep Planning Domain Learning”. In: *2020 IEEE International Conference on Robotics and Automation (ICRA)* (2020), pp. 10459–10465.

[155] Maurice George Kendall. “Rank correlation methods.” In: (1948).

[156] Diederik P. Kingma and Max Welling. “Auto-Encoding Variational Bayes”. In: *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*. Ed. by Yoshua Bengio and Yann LeCun. 2014. URL: http://arxiv.org/abs/1312.6114.

[157] Kris M. Kitani et al. “Activity Forecasting”. In: *Computer Vision – ECCV 2012*. Ed. by Andrew Fitzgibbon et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 201–214. ISBN: 978-3-642-33765-9.

[158] Heather Knight and Reid Simmons. “Expressive motion with x, y and theta: Laban Effort Features for mobile robots”. In: *The 23rd IEEE International Symposium on Robot and Human Interactive Communication*. 2014, pp. 267–273. DOI: 10.1109/ROMAN.2014.6926264.

[159] Marin Kobilarov. “Cross-Entropy Randomized Motion Planning”. In: June 2011. DOI: 10.15607/RSS.2011.VII.022.

[160] Nathan P. Koenig and Maja J. Mataric. “Robot life-long task learning from human demonstrations: a Bayesian approach”. In: *Auton. Robots* 41.5 (2017), pp. 1173–1188. DOI: 10.1007/s10514-016-9601-1. URL: https://doi.org/10.1007/s10514-016-9601-1.

[161] J Zico Kolter et al. “A probabilistic approach to mixed open-loop and closed-loop control, with application to extreme autonomous driving”. In: *2010 IEEE International Conference on Robotics and Automation*. IEEE. 2010, pp. 839–845.

[162] Florian Köpf et al. "Inverse Reinforcement Learning for Identification in Linear-Quadratic Dynamic Games". In: *IFAC-PapersOnLine* 50.1 (2017). 20th IFAC World Congress, pp. 14902–14908. ISSN: 2405-8963. DOI: https://doi.org/10.1016/j.ifacol.2017.08.2537. URL: http://www.sciencedirect.com/science/article/pii/S2405896317334596.

[163] Donald H. Kraft. "A software package for sequential quadratic programming". In: 1988.

[164] Henrik Kretzschmar et al. "Socially Compliant Mobile Robot Navigation via Inverse Reinforcement Learning". In: *Int. J. Rob. Res.* 35.11 (Sept. 2016), pp. 1289–1307. ISSN: 0278-3649. DOI: 10.1177/0278364915619772. URL: https://doi.org/10.1177/0278364915619772.

[165] Markus Kuderer et al. "Feature-Based Prediction of Trajectories for Socially Compliant Navigation." In.

[166] J.J. Kuffner and S.M. LaValle. "RRT-connect: An efficient approach to single-query path planning". In: *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065)*. Vol. 2. 2000, 995–1001 vol.2. DOI: 10.1109/ROBOT.2000.844730.

[167] Johannes Kulick et al. "Active Learning for Teaching a Robot Grounded Relational Symbols". In: *IJCAI 2013, Proceedings of the 23rd International Joint Conference on Artificial Intelligence, Beijing, China, August 3-9, 2013*. Ed. by Francesca Rossi. IJCAI/AAAI, 2013, pp. 1451–1457. URL: http://www.aaai.org/ocs/index.php/IJCAI/IJCAI13/paper/view/6706.

[168] Minae Kwon, Sandy H. Huang, and Anca D. Dragan. "Expressing Robot Incapability". In: *Proceedings of the 2018 ACM/IEEE International Conference on Human-Robot Interaction*. HRI '18. Chicago, IL, USA: Association for Computing Machinery, 2018, pp. 87–95. ISBN: 9781450349536. DOI: 10.1145/3171221.3171276.

[169] Cheng-I Lai. "Contrastive Predictive Coding Based Feature for Automatic Speaker Verification". In: *arXiv preprint arXiv:1904.01575* (2019).

[170] Michael Laskin, Aravind Srinivas, and Pieter Abbeel. "CURL: Contrastive Unsupervised Representations for Reinforcement Learning". In: *Proceedings of the 37th International Conference on Machine Learning*. Ed. by Hal Daumé III and Aarti Singh. Vol. 119. Proceedings of Machine Learning Research. PMLR, 2020, pp. 5639–5650. URL: https://proceedings.mlr.press/v119/laskin20a.html.

[171] Kimin Lee, Laura M. Smith, and Pieter Abbeel. "PEBBLE: Feedback-Efficient Interactive Reinforcement Learning via Relabeling Experience and Unsupervised Pre-training". In: *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*. Ed. by Marina Meila and Tong Zhang. Vol. 139. Proceedings of Machine Learning Research. PMLR, 2021, pp. 6152–6163. URL: http://proceedings.mlr.press/v139/lee21i.html.

[172] Seong Jae Lee and Zoran Popović. "Learning Behavior Styles with Inverse Reinforcement Learning". In: *ACM SIGGRAPH 2010 Papers*. SIGGRAPH '10. Los Angeles, California: Association for Computing Machinery, 2010. ISBN: 9781450302104. DOI: 10.1145/1833349.1778859.

[173] Emmanuel Lesaffre. "Superiority, equivalence, and non-inferiority trials". In: *Bulletin of the NYU hospital for joint diseases* 66.2 (2008), pp. 150–154. ISSN: 1936-9719. URL: http://europepmc.org/abstract/MED/18537788.

[174] Timothée Lesort et al. "State representation learning for control: An overview". In: *Neural Networks* 108 (2018), pp. 379–392. ISSN: 0893-6080. DOI: https://doi.org/10.1016/j.neunet.2018.07.006. URL: https://www.sciencedirect.com/science/article/pii/S0893608018302053.

[175] Sergey Levine and Vladlen Koltun. "Continuous Inverse Optimal Control with Locally Optimal Examples". In: *Proceedings of the 29th International Coference on International Conference on Machine Learning*. ICML'12. Edinburgh, Scotland: Omnipress, 2012, pp. 475–482. ISBN: 978-1-4503-1285-1. URL: http://dl.acm.org/citation.cfm?id=3042573.3042637.

[176] Sergey Levine and Vladlen Koltun. "Continuous Inverse Optimal Control with Locally Optimal Examples". In: *ArXiv* abs/1206.4617 (2012).

[177] Sergey Levine, Zoran Popovic, and Vladlen Koltun. "Feature construction for inverse reinforcement learning". In: *Advances in Neural Information Processing Systems*. 2010, pp. 1342–1350.

[178] Sergey Levine, Zoran Popovic, and Vladlen Koltun. "Nonlinear inverse reinforcement learning with gaussian processes". In: *Advances in Neural Information Processing Systems*. 2011, pp. 19–27.

[179] Sergey Levine et al. "Offline reinforcement learning: Tutorial, review, and perspectives on open problems". In: *arXiv preprint arXiv:2005.01643* (2020).

[180] Kejun Li et al. "Roial: Region of interest active learning for characterizing exoskeleton gait preference landscapes". In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2021, pp. 3212–3218.

[181] Qinan Li, Weidong Chen, and Jingchuan Wang. "Dynamic shared control for human-wheelchair cooperation". In: *2011 IEEE International Conference on Robotics and Automation*. IEEE. 2011, pp. 4278–4283.

[182] Zhongyu Li, Christine Cummings, and Koushil Sreenath. "Animated Cassie: A Dynamic Relatable Robotic Character". In: *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (2020), pp. 3739–3746.

[183] Zhongyu Li et al. "Vision-Aided Autonomous Navigation of Bipedal Robots in Height-Constrained Environments". In: *arXiv preprint arXiv:2109.05714* (2021).

[184] Hun-Ok Lim, A. Ishii, and A. Takanishi. "Basic emotional walking using a biped humanoid robot". In: *IEEE SMC'99 Conference Proceedings. 1999 IEEE International Conference on Systems, Man, and Cybernetics (Cat. No.99CH37028)*. Vol. 4. 1999, 954–959 vol.4. DOI: 10.1109/ICSMC.1999.812539.

[185] Long-Ji Lin. "Hierarchical learning of robot skills by reinforcement". In: *Proceedings of International Conference on Neural Networks (ICNN'88), San Francisco, CA, USA, March 28 - April 1, 1993*. IEEE, 1993, pp. 181–186. DOI: 10.1109/ICNN.1993.298553. URL: https://doi.org/10.1109/ICNN.1993.298553.

[186] C. Karen Liu, Aaron Hertzmann, and Zoran Popović. "Learning Physics-Based Motion Style with Nonlinear Inverse Optimization". In: *ACM Trans. Graph.* 24.3 (July 2005), pp. 1071–1081. ISSN: 0730-0301. DOI: 10.1145/1073204.1073314.

[187] Weiyu Liu. "A survey of semantic reasoning frameworks for robotic systems". In: (2022). URL: http://weiyuliu.com/data/A_Survey_of_Semantic_Reasoning_Frameworks_for_Robotic_Systems.pdf.

[188] Manuel Lopes, Francisco Melo, and Luis Montesano. "Active learning for reward estimation in inverse reinforcement learning". In: *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer. 2009, pp. 31–46.

[189] D. P. Losey et al. "Controlling Assistive Robots with Learned Latent Actions". In: *2020 IEEE International Conference on Robotics and Automation (ICRA)*. 2020, pp. 378–384. DOI: 10.1109/ICRA40945.2020.9197197.

[190] Dylan P Losey and Marcia K O'Malley. "Including uncertainty when learning from human corrections". In: *arXiv preprint arXiv:1806.02454* (2018).

[191] Dylan P Losey and Marcia K O'Malley. "Trajectory deformations from physical human–robot interaction". In: *IEEE Transactions on Robotics* 34.1 (2017), pp. 126–138.

[192] Dylan P. Losey et al. "A Review of Intent Detection, Arbitration, and Communication Aspects of Shared Control for Physical Human–Robot Interaction". In: *Applied Mechanics Reviews* 70.1 (Feb. 2018). 010804. ISSN: 0003-6900. DOI: 10.1115/1.4039145. URL: https://doi.org/10.1115/1.4039145.

[193] Christos Louizos et al. "The Variational Fair Autoencoder". In: *CoRR* abs/1511.00830 (2016).

[194] R. Duncan Luce. *Individual choice behavior.* Oxford, England: John Wiley, 1959, pp. xii, 153–xii, 153.

[195] R.Duncan Luce. "The choice axiom after twenty years". In: *Journal of Mathematical Psychology* 15.3 (1977), pp. 215–233. ISSN: 0022-2496. DOI: https://doi.org/10.1016/0022-2496(77)90032-3. URL: http://www.sciencedirect.com/science/article/pii/0022249677900323.

[196] Hoai Luu-Duc and Jun Miura. “An Incremental Feature Set Refinement in a Programming by Demonstration Scenario”. In: *4th IEEE International Conference on Advanced Robotics and Mechatronics, ICARM 2019, Toyonaka, Japan, July 3-5, 2019*. IEEE, 2019, pp. 372–377. DOI: 10.1109/ICARM.2019.8833723. URL: https://doi.org/10.1109/ICARM.2019.8833723.

[197] Corey Lynch et al. “Learning Latent Plans from Play”. In: *3rd Annual Conference on Robot Learning, CoRL 2019, Osaka, Japan, October 30 - November 1, 2019, Proceedings*. Ed. by Leslie Pack Kaelbling, Danica Kragic, and Komei Sugiura. Vol. 100. Proceedings of Machine Learning Research. PMLR, 2019, pp. 1113–1132. URL: http://proceedings.mlr.press/v100/lynch20a.html.

[198] J. Mainprice and D. Berenson. “Human-robot collaborative manipulation planning using early prediction of human motion”. In: *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2013, pp. 299–306. DOI: 10.1109/IROS.2013.6696368.

[199] J. Mainprice, R. Hayne, and D. Berenson. “Predicting human reaching motion in collaborative tasks using Inverse Optimal Control and iterative re-planning”. In: *2015 IEEE International Conference on Robotics and Automation (ICRA)*. May 2015, pp. 885–892. DOI: 10.1109/ICRA.2015.7139282.

[200] Viktor Makoviychuk et al. *Isaac Gym: High Performance GPU-Based Physics Simulation For Robot Learning*. 2021. arXiv: 2108.10470 [cs.RO].

[201] Zhao Mandi, Pieter Abbeel, and Stephen James. “On the Effectiveness of Finetuning Versus Meta-reinforcement Learning”. In: *arXiv preprint arXiv:2206.03271* (2022).

[202] Aarian Marshal and Alex Davies. “Uber’s Self-Driving Car Didn’t Know Pedestrians Could Jaywalk”. In: *Wired* (Nov. 2019). URL: https://www.wired.com/story/ubers-self-driving-car-didnt-know-pedestrians-could-jaywalk/.

[203] Brian McFee, Gert Lanckriet, and Tony Jebara. “Learning Multi-modal Similarity.” In: *Journal of machine learning research* 12.2 (2011).

[204] Oier Mees et al. “Metric learning for generalizing spatial relations to new objects”. In: *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2017, pp. 3175–3182. DOI: 10.1109/IROS.2017.8206149.

[205] Saif M. Mohammad. “Obtaining Reliable Human Ratings of Valence, Arousal, and Dominance for 20,000 English Words”. In: *Proceedings of The Annual Conference of the Association for Computational Linguistics (ACL)*. Melbourne, Australia, 2018.

[206] Anahita Mohseni-Kabir et al. “Interactive Hierarchical Task Learning from a Single Demonstration”. In: *Proceedings of the Tenth Annual ACM/IEEE International Conference on Human-Robot Interaction, HRI 2015, Portland, OR, USA, March 2-5, 2015*. Ed. by Julie A. Adams et al. ACM, 2015, pp. 205–212. DOI: 10.1145/2696454.2696474. URL: https://doi.org/10.1145/2696454.2696474.

[207] Anahita Mohseni-Kabir et al. “Simultaneous learning of hierarchy and primitives for complex robot tasks”. In: *Autonomous Robots* 43.4 (2019), pp. 859–874.

[208] Tiago Mota and Mohan Sridharan. “Incrementally Grounding Expressions for Spatial Relations between Objects”. In: *Proceedings of the 27th International Joint Conference on Artificial Intelligence*. IJCAI’18. Stockholm, Sweden: AAAI Press, 2018, pp. 1928–1934. ISBN: 9780999241127.

[209] Arsalan Mousavian, Clemens Eppner, and Dieter Fox. “6-DOF GraspNet: Variational Grasp Generation for Object Manipulation”. In: *International Conference on Computer Vision (ICCV)*. 2019.

[210] Katharina Muelling et al. “Autonomy infused teleoperation with application to brain computer interface controlled manipulation”. In: *Autonomous Robots* 41.6 (2017), pp. 1401–1422.

[211] Shohin Mukherjee et al. “Reactive Long Horizon Task Execution via Visual Skill and Precondition Models”. In: *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2020, pp. 5717–5724.

[212] Jonathan Mumm and Bilge Mutlu. “Human-robot proxemics: physical and psychological distancing in human-robot interaction”. In: *Proceedings of the 6th international conference on Human-robot interaction*. 2011, pp. 331–338.

[213] Negin Nejati, Pat Langley, and Tolga Könik. “Learning hierarchical task networks by observation”. In: *Machine Learning, Proceedings of the Twenty-Third International Conference (ICML 2006), Pittsburgh, Pennsylvania, USA, June 25-29, 2006*. Ed. by William W. Cohen and Andrew W. Moore. Vol. 148. ACM International Conference Proceeding Series. ACM, 2006, pp. 665–672. DOI: 10.1145/1143844.1143928. URL: https://doi.org/10.1145/1143844.1143928.

[214] Andrew Ng and Stuart Russell. “Algorithms for inverse reinforcement learning”. In: *International Conference on Machine Learning (ICML)* 0 (2000), pp. 663–670. ISSN: 00029645. DOI: 10.2460/ajvr.67.2.323. URL: http://www-cs.stanford.edu/people/ang/papers/icml00-irl.pdf.

[215] Kentaro Nishi and Masamichi Shimosaka. “Fine-grained driving behavior prediction via context-aware multi-task inverse reinforcement learning”. In: *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2020, pp. 2281–2287.

[216] Daniel Nyga et al. “Grounding Robot Plans from Natural Language Instructions with Incomplete World Knowledge”. In: *2nd Annual Conference on Robot Learning, CoRL 2018, Zürich, Switzerland, 29-31 October 2018, Proceedings*. Vol. 87. Proceedings of Machine Learning Research. PMLR, 2018, pp. 714–723. URL: http://proceedings.mlr.press/v87/nyga18a.html.

[217] Oliver Obst. “Using a Planner for Coordination of Multiagent Team Behavior”. In: *Programming Multi-Agent Systems, Third International Workshop, ProMAS 2005, Utrecht, The Netherlands, July 26, 2005, Revised and Invited Papers*. Ed. by Rafael H. Bordini et al. Vol. 3862. Lecture Notes in Computer Science. Springer, 2005, pp. 90–100. DOI: `10.1007/11678823\_6`. URL: `https://doi.org/10.1007/11678823%5C_6`.

[218] Aäron van den Oord, Yazhe Li, and Oriol Vinyals. “Representation Learning with Contrastive Predictive Coding”. In: *CoRR* abs/1807.03748 (2018). arXiv: `1807.03748`. URL: `http://arxiv.org/abs/1807.03748`.

[219] Takayuki Osa et al. “An Algorithmic Perspective on Imitation Learning”. In: *Foundations and Trends in Robotics* 7.1-2 (2018), pp. 1–179.

[220] Alexandros Paraschos et al. “Probabilistic Movement Primitives”. In: *Advances in Neural Information Processing Systems 26*. Ed. by C. J. C. Burges et al. Curran Associates, Inc., 2013, pp. 2616–2624. URL: `http://papers.nips.cc/paper/5177-probabilistic-movement-primitives.pdf`.

[221] Adam Paszke et al. “PyTorch: An Imperative Style, High-Performance Deep Learning Library”. In: *NeurIPS*. 2019.

[222] Deepak Pathak et al. “Zero-Shot Visual Imitation”. In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. 2018, pp. 2131–21313. DOI: `10.1109/CVPRW.2018.00278`.

[223] Abhishek Paudel. “Learning for Robot Decision Making under Distribution Shift: A Survey”. In: *arXiv preprint arXiv:2203.07558* (2022).

[224] Chris Paxton et al. “Predicting Stable Configurations for Semantic Placement of Novel Objects”. In: *Conference on Robot Learning (CoRL)*. to appear. 2021.

[225] Claudia Pérez-D’Arpino and Julie A Shah. “Fast target prediction of human reaching motion for cooperative human-robot manipulation tasks using time series classification”. In: *2015 IEEE international conference on robotics and automation (ICRA)*. IEEE. 2015, pp. 6175–6182.

[226] M. Pfeiffer et al. “Predicting actions to act predictably: Cooperative partial motion planning with maximum entropy models”. In: *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2016, pp. 2096–2101. DOI: `10.1109/IROS.2016.7759329`.

[227] C. Qi et al. “PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation”. In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2017), pp. 77–85.

[228] Charles Ruizhongtai Qi et al. “Pointnet++: Deep hierarchical feature learning on point sets in a metric space”. In: *Advances in Neural Information Processing Systems*. 2017, pp. 5099–5108.

[229] Alec Radford et al. "Learning transferable visual models from natural language supervision". In: *International Conference on Machine Learning*. PMLR. 2021, pp. 8748–8763.

[230] Rouhollah Rahmatizadeh et al. "Vision-Based Multi-Task Manipulation for Inexpensive Robots Using End-to-End Learning from Demonstration". In: *2018 IEEE International Conference on Robotics and Automation, ICRA 2018, Brisbane, Australia, May 21-25, 2018*. IEEE, 2018, pp. 3758–3765. DOI: 10.1109/ICRA.2018.8461076. URL: https://doi.org/10.1109/ICRA.2018.8461076.

[231] Aravind Rajeswaran et al. "Learning Complex Dexterous Manipulation with Deep Reinforcement Learning and Demonstrations". In: *Robotics: Science and Systems XIV, Carnegie Mellon University, Pittsburgh, Pennsylvania, USA, June 26-30, 2018*. Ed. by Hadas Kress-Gazit et al. 2018. DOI: 10.15607/RSS.2018.XIV.049. URL: http://www.roboticsproceedings.org/rss14/p49.html.

[232] Deepak Ramachandran and Eyal Amir. "Bayesian Inverse Reinforcement Learning". In: *Proceedings of the 20th International Joint Conference on Artifical Intelligence*. IJCAI'07. Hyderabad, India: Morgan Kaufmann Publishers Inc., 2007, pp. 2586–2591. URL: http://dl.acm.org/citation.cfm?id=1625275.1625692.

[233] Deepak Ramachandran and Eyal Amir. "Bayesian Inverse Reinforcement Learning." In: *IJCAI*. Vol. 7. 2007, pp. 2586–2591.

[234] Aditya Ramesh et al. "Zero-shot text-to-image generation". In: *International Conference on Machine Learning*. PMLR. 2021, pp. 8821–8831.

[235] Nathan Ratliff et al. "Boosting structured prediction for imitation learning". In: *Advances in Neural Information Processing Systems*. 2007, pp. 1153–1160.

[236] Nathan D Ratliff, J Andrew Bagnell, and Martin A Zinkevich. "Maximum margin planning". In: *Proceedings of the 23rd international conference on Machine learning*. ACM. 2006, pp. 729–736.

[237] Nathan D. Ratliff, J. Andrew Bagnell, and Martin A. Zinkevich. "Maximum Margin Planning". In: *Proceedings of the 23rd International Conference on Machine Learning*. ICML '06. Pittsburgh, Pennsylvania, USA: Association for Computing Machinery, 2006, pp. 729–736. ISBN: 1595933832. DOI: 10.1145/1143844.1143936. URL: https://doi.org/10.1145/1143844.1143936.

[238] S. Reddy et al. "Learning Human Objectives by Evaluating Hypothetical Behavior". In: *ICML*. 2020.

[239] Sid Reddy, Anca D. Dragan, and Sergey Levine. "Pragmatic Image Compression for Human-in-the-Loop Decision-Making". In: *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*. Ed. by Marc'Aurelio Ranzato et al. 2021, pp. 26499–26510. URL: https://proceedings.neurips.cc/paper/2021/hash/df0aab058ce179e4f7ab135ed4e641a9-Abstract.html.

[240] Siddharth Reddy, Anca D Dragan, and Sergey Levine. “Shared autonomy via deep reinforcement learning”. In: *arXiv preprint arXiv:1802.01744* (2018).

[241] Siddharth Reddy, Anca D. Dragan, and Sergey Levine. “SQIL: Imitation Learning via Reinforcement Learning with Sparse Rewards”. In: *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. URL: https://openreview.net/forum?id=S1xKd24twB.

[242] Siddharth Reddy et al. “Learning Human Objectives by Evaluating Hypothetical Behavior”. In: *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*. Vol. 119. Proceedings of Machine Learning Research. PMLR, 2020, pp. 8020–8029. URL: http://proceedings.mlr.press/v119/reddy20a.html.

[243] C. J. Reed et al. “Self-Supervised Pretraining Improves Self-Supervised Pretraining”. In: *2022 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*. Los Alamitos, CA, USA: IEEE Computer Society, 2022, pp. 1050–1060. DOI: 10.1109/WACV51458.2022.00112. URL: https://doi.ieeecomputersociety.org/10.1109/WACV51458.2022.00112.

[244] Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. “A reduction of imitation learning and structured prediction to no-regret online learning”. In: *Proceedings of the fourteenth international conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings. 2011, pp. 627–635.

[245] Cynthia Rudin. “Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead”. In: *Nature Machine Intelligence* 1.5 (2019), pp. 206–215.

[246] James A. Russell. “A circumplex model of affect”. In: *Journal of Personality and Social Psychology* 39.6 (Dec. 1980), pp. 1161–1178. DOI: 10.1037/h0077714.

[247] Stuart Russell and Peter Norvig. “Artificial intelligence: a modern approach”. In: (2002).

[248] Stuart J Russell. *Artificial intelligence a modern approach*. Pearson Education, Inc., 2010.

[249] D. Sadigh et al. “Information gathering actions over human internal state”. In: *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Oct. 2016, pp. 66–73. DOI: 10.1109/IROS.2016.7759036.

[250] Dorsa Sadigh et al. “Active Preference-Based Learning of Reward Functions”. In: *Robotics: Science and Systems*. 2017.

[251] Dorsa Sadigh et al. “Active preference-based learning of reward functions”. In: *Robotics: Science and systems*. 2017.

[252] Martin Saerbeck and Christoph Bartneck. “Perception of affect elicited by robot motion”. In: *2010 5th ACM/IEEE International Conference on Human-Robot Interaction (HRI)* (2010), pp. 53–60.

[253] Scott Sanner. “Simultaneous learning of structure and value in relational reinforcement learning”. In: *Workshop on Rich Representations for Reinforcement Learning*. Citeseer. 2005, p. 57.

[254] Ashutosh Saxena et al. “RoboBrain: Large-Scale Knowledge Engine for Robots”. In: *CoRR* abs/1412.0691 (2014). arXiv: 1412.0691. URL: http://arxiv.org/abs/1412.0691.

[255] John Schulman et al. “Finding Locally Optimal, Collision-Free Trajectories with Sequential Convex Optimization.” In: *Robotics: science and systems*. Vol. 9. 1. Citeseer. 2013, pp. 1–10.

[256] Max Schwarzer et al. “Pretraining Representations for Data-Efficient Reinforcement Learning”. In: *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*. Ed. by Marc’Aurelio Ranzato et al. 2021. URL: https://proceedings.neurips.cc/paper/2021/hash/69eba34671b3ef1ef38ee85caae6b2a1-Abstract.html.

[257] Seyed Kamyar Seyed Ghasemipour, Shixiang Shane Gu, and Richard Zemel. “Smile: Scalable meta inverse reinforcement learning through context-conditional policies”. In: *Advances in Neural Information Processing Systems* 32 (2019).

[258] Rohin Shah et al. “The Implicit Preference Information in an Initial State”. In: *International Conference on Learning Representations*. 2019. URL: https://openreview.net/forum?id=rkevMnRqYQ.

[259] Lin Shao et al. “Concept2Robot: Learning Manipulation Concepts from Instructions and Human Demonstrations”. In: *Robotics: Science and Systems*. 2020.

[260] Megha Sharma et al. “Communicating Affect via Flight Path: Exploring Use of the Laban Effort System for Designing Affective Locomotion Paths”. In: *Proceedings of the 8th ACM/IEEE International Conference on Human-Robot Interaction*. HRI ’13. Tokyo, Japan: IEEE Press, 2013, pp. 293–300. ISBN: 9781467330558.

[261] Roger N. Shepard. “Stimulus and response generalization: A stochastic model relating generalization to distance in psychological space”. In: *Psychometrika* 22.4 (1957), pp. 325–345. ISSN: 1860-0980. DOI: 10.1007/BF02288967. URL: https://doi.org/10.1007/BF02288967.

[262] Pannaga Shivaswamy and Thorsten Joachims. “Coactive learning”. In: *Journal of Artificial Intelligence Research* 53 (2015), pp. 1–40.

[263] Avi Singh et al. "End-To-End Robotic Reinforcement Learning without Reward Engineering". In: *Robotics: Science and Systems XV, University of Freiburg, Freiburg im Breisgau, Germany, June 22-26, 2019*. Ed. by Antonio Bicchi, Hadas Kress-Gazit, and Seth Hutchinson. 2019. DOI: 10.15607/RSS.2019.XV.073. URL: https://doi.org/10.15607/RSS.2019.XV.073.

[264] Avi Singh et al. "Scalable Multi-Task Imitation Learning with Autonomous Improvement". In: *2020 IEEE International Conference on Robotics and Automation, ICRA 2020, Paris, France, May 31 - August 31, 2020*. IEEE, 2020, pp. 2167–2173. DOI: 10.1109/ICRA40945.2020.9197020. URL: https://doi.org/10.1109/ICRA40945.2020.9197020.

[265] Dan Song et al. "Multivariate discretization for Bayesian Network structure learning in robot grasping". In: *IEEE International Conference on Robotics and Automation, ICRA 2011, Shanghai, China, 9-13 May 2011*. IEEE, 2011, pp. 1944–1950. DOI: 10.1109/ICRA.2011.5979666. URL: https://doi.org/10.1109/ICRA.2011.5979666.

[266] Arjun Sripathy et al. *Teaching Robots to Span the Space of Functional Expressive Motion*. 2022. DOI: 10.48550/ARXIV.2203.02091. URL: https://arxiv.org/abs/2203.02091.

[267] Arjun Sripathy et al. *Teaching Robots to Span the Space of Functional Expressive Motion*. 2022. DOI: 10.48550/ARXIV.2203.02091. URL: https://arxiv.org/abs/2203.02091.

[268] Neil Stewart, Gordon DA Brown, and Nick Chater. "Absolute identification by relative judgment." In: *Psychological review* 112.4 (2005), p. 881.

[269] Adam Stooke et al. "Decoupling Representation Learning from Reinforcement Learning". In: *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*. Ed. by Marina Meila and Tong Zhang. Vol. 139. Proceedings of Machine Learning Research. PMLR, 2021, pp. 9870–9879. URL: http://proceedings.mlr.press/v139/stooke21a.html.

[270] Michael Suguitan, Randy Gomez, and Guy Hoffman. "Demonstrating MoveAE: Modifying Affective Robot Movements Using Classifying Variational Autoencoders". In: *Companion of the 2020 ACM/IEEE International Conference on Human-Robot Interaction* (2020).

[271] Martin Sundermeyer et al. "Implicit 3D Orientation Learning for 6D Object Detection from RGB Images". In: *ECCV*. 2018.

[272] Daniel Szafir, Bilge Mutlu, and Terrence Fong. "Communication of Intent in Assistive Free Flyers". In: HRI '14. Bielefeld, Germany: Association for Computing Machinery, 2014, pp. 358–365. ISBN: 9781450326582. DOI: 10.1145/2559636.2559672.

[273] Leila Takayama and Doug Dooley. "Expressing thought: Improving robot readability with animation principles". In: Jan. 2011, pp. 69–76. DOI: 10.1145/1957656.1957674.

[274] Omer Tamuz et al. "Adaptively learning the crowd kernel". In: *arXiv preprint arXiv:1105.1033* (2011).

[275] Akshaya Thippur et al. "A Comparison of Qualitative and Metric Spatial Relation Models for Scene Understanding". In: *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*. AAAI'15. Austin, Texas: AAAI Press, 2015, pp. 1632–1640. ISBN: 0262511290.

[276] Faraz Torabi, Garrett Warnell, and Peter Stone. "Behavioral Cloning from Observation". In: *Proceedings of the 27th International Joint Conference on Artificial Intelligence*. IJCAI'18. Stockholm, Sweden: AAAI Press, 2018, pp. 4950–4957. ISBN: 9780999241127.

[277] Lorenzo Torresani, Peggy Hackney, and Christoph Bregler. "Learning Motion Style Synthesis from Perceptual Observations." In: Jan. 2006, pp. 1393–1400.

[278] Mycal Tucker, Yilun Zhou, and Julie Shah. "Latent Space Alignment Using Adversarially Guided Self-Play". In: *International Journal of Human–Computer Interaction* 0.0 (2022), pp. 1–19. DOI: 10.1080/10447318.2022.2083463. eprint: https://doi.org/10.1080/10447318.2022.2083463. URL: https://doi.org/10.1080/10447318.2022.2083463.

[279] Vladimir Vapnik. *The nature of statistical learning theory*. Springer science & business media, 2013.

[280] D. Vasquez, B. Okal, and K. O. Arras. "Inverse Reinforcement Learning algorithms and features for robot navigation in crowds: An experimental comparison". In: *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2014, pp. 1341–1346. DOI: 10.1109/IROS.2014.6942731.

[281] Paul Vernaza and Drew Bagnell. "Efficient high dimensional maximum entropy modeling via symmetric partition functions". In: *Advances in Neural Information Processing Systems*. 2012, pp. 575–583.

[282] John Von Neumann and Oskar Morgenstern. *Theory of games and economic behavior*. Princeton University Press Princeton, NJ, 1945.

[283] Peter Vovsha. "Application of Cross-Nested Logit Model to Mode Choice in Tel Aviv, Israel, Metropolitan Area". In: *Transportation Research Record* 1607.1 (1997), pp. 6–15. DOI: 10.3141/1607-02. eprint: https://doi.org/10.3141/1607-02. URL: https://doi.org/10.3141/1607-02.

[284] Chen Wang et al. "6-PACK: Category-level 6D Pose Tracker with Anchor-Based Keypoints". In: *2020 IEEE International Conference on Robotics and Automation (ICRA)*. 2020, pp. 10059–10066. DOI: 10.1109/ICRA40945.2020.9196679.

[285] Quan Wang et al. "Knowledge graph embedding: A survey of approaches and applications". In: *IEEE Transactions on Knowledge and Data Engineering* 29.12 (2017), pp. 2724–2743.

[286] Garrett Warnell et al. “Deep TAMER: Interactive Agent Shaping in High-Dimensional State Spaces”. In: *ArXiv* abs/1709.10163 (2018).

[287] Manuel Watter et al. “Embed to Control: A Locally Linear Latent Dynamics Model for Control from Raw Images”. In: *Advances in Neural Information Processing Systems*. Vol. 28. Curran Associates, Inc., 2015. URL: https://proceedings.neurips.cc/paper/2015/file/a1afc58c6ca9540d057299ec3016d726-Paper.pdf.

[288] Stefan Wellek. *Testing statistical hypotheses of equivalence and noninferiority*. Chapman and Hall/CRC, 2010.

[289] Erik Wijmans. “Pointnet++ Pytorch”. In: *https://github.com/erikwijmans/Pointnet2_PyTorch* (2018).

[290] Christian Wirth et al. “A survey of preference-based reinforcement learning methods”. In: *Journal of Machine Learning Research* 18.136 (2017), pp. 1–46.

[291] M. Wulfmeier, D. Z. Wang, and I. Posner. “Watch this: Scalable cost-function learning for path planning in urban environments”. In: *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2016, pp. 2089–2095.

[292] Shihong Xia et al. “Realtime Style Transfer for Unlabeled Heterogeneous Human Motion”. In: *ACM Trans. Graph.* 34.4 (July 2015). ISSN: 0730-0301. DOI: 10.1145/2766999.

[293] Yikun Xian et al. “Reinforcement Knowledge Graph Reasoning for Explainable Recommendation”. In: *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR’19. Paris, France: Association for Computing Machinery, 2019, pp. 285–294. ISBN: 9781450361729. DOI: 10.1145/3331184.3331203. URL: https://doi.org/10.1145/3331184.3331203.

[294] Yu Xiang et al. “Learning RGB-D Feature Embeddings for Unseen Object Instance Segmentation”. In: *Conference on Robot Learning (CoRL)*. 2020.

[295] Yu Xiang et al. “PoseCNN: A Convolutional Neural Network for 6D Object Pose Estimation in Cluttered Scenes”. In: June 2018. DOI: 10.15607/RSS.2018.XIV.019.

[296] Kelvin Xu et al. “Learning a Prior over Intent via Meta-Inverse Reinforcement Learning”. In: *Proceedings of the 36th International Conference on Machine Learning*. Ed. by Kamalika Chaudhuri and Ruslan Salakhutdinov. Vol. 97. Proceedings of Machine Learning Research. PMLR, June 2019, pp. 6952–6962. URL: https://proceedings.mlr.press/v97/xu19d.html.

[297] Kelvin Xu et al. “Learning a prior over intent via meta-inverse reinforcement learning”. In: *International Conference on Machine Learning*. PMLR. 2019, pp. 6952–6962.

[298] Jun Yamada et al. “Task-Induced Representation Learning”. In: *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022. URL: https://openreview.net/forum?id=OzyXtIZAzFv.

[299] Mengjiao Yang and Ofir Nachum. “Representation Matters: Offline Pretraining for Sequential Decision Making”. In: *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*. Ed. by Marina Meila and Tong Zhang. Vol. 139. Proceedings of Machine Learning Research. PMLR, 2021, pp. 11784–11794. URL: http://proceedings.mlr.press/v139/yang21h.html.

[300] Wei Yang et al. “Reactive human-to-robot handovers of arbitrary objects”. In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2021.

[301] John Seon Keun Yi, Yoonwoo Kim, and Sonia Chernova. “Incremental Object Grounding Using Scene Graphs”. In: *CoRR* abs/2201.01901 (2022). arXiv: 2201.01901. URL: https://arxiv.org/abs/2201.01901.

[302] Lantao Yu et al. “Meta-inverse reinforcement learning with probabilistic context variables”. In: *Advances in Neural Information Processing Systems* 32 (2019).

[303] Wentao Yuan et al. “SORNet: Spatial Object-Centric Representations for Sequential Manipulation”. In: *5th Annual Conference on Robot Learning*. PMLR. 2021, pp. 148–157.

[304] Alireza Zareian, Svebor Karaman, and Shih-Fu Chang. “Bridging Knowledge Graphs to Generate Scene Graphs”. In: *Computer Vision - ECCV 2020 - 16th European Conference, Glasgow, UK, August 23-28, 2020, Proceedings, Part XXIII*. Ed. by Andrea Vedaldi et al. Vol. 12368. Lecture Notes in Computer Science. Springer, 2020, pp. 606–623. DOI: 10.1007/978-3-030-58592-1_36. URL: https://doi.org/10.1007/978-3-030-58592-1%5C_36.

[305] Amy Zhang et al. “Learning Invariant Representations for Reinforcement Learning without Reconstruction”. In: *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. URL: https://openreview.net/forum?id=-2FCwDKRREu.

[306] Tianhao Zhang et al. “Deep Imitation Learning for Complex Manipulation Tasks from Virtual Reality Teleoperation”. In: *2018 IEEE International Conference on Robotics and Automation, ICRA 2018, Brisbane, Australia, May 21-25, 2018*. IEEE, 2018, pp. 1–8. DOI: 10.1109/ICRA.2018.8461249. URL: https://doi.org/10.1109/ICRA.2018.8461249.

[307] Yichuan Zhang et al. “Efficient Reinforcement Learning from Demonstration via Bayesian Network-Based Knowledge Extraction”. In: *Comput. Intell. Neurosci.* 2021 (2021), 7588221:1–7588221:16. DOI: 10.1155/2021/7588221. URL: https://doi.org/10.1155/2021/7588221.

[308] Fedor Zhdanov. “Diverse mini-batch Active Learning”. In: *ArXiv* abs/1901.05954 (2019).

[309] Jiangchuan Zheng, Siyuan Liu, and Lionel M. Ni. "Robust Bayesian Inverse Reinforcement Learning with Sparse Behavior Noise". In: *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*. AAAI'14. Quebec City, Quebec, Canada: AAAI Press, 2014, pp. 2198–2205. URL: http://dl.acm.org/citation.cfm?id=2892753.2892857.

[310] Allan Zhou and Anca D. Dragan. "Cost Functions for Robot Motion Style". In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (2018), pp. 3632–3639.

[311] Brian D. Ziebart et al. "Maximum Entropy Inverse Reinforcement Learning". In: *Proceedings of the 23rd National Conference on Artificial Intelligence - Volume 3*. AAAI'08. Chicago, Illinois: AAAI Press, 2008, pp. 1433–1438. ISBN: 978-1-57735-368-3. URL: http://dl.acm.org/citation.cfm?id=1620270.1620297.

[312] Brian D. Ziebart et al. "Planning-based Prediction for Pedestrians". In: *Proceedings of the 2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IROS'09. St. Louis, MO, USA: IEEE Press, 2009, pp. 3931–3936. ISBN: 978-1-4244-3803-7. URL: http://dl.acm.org/citation.cfm?id=1732643.1732694.

[313] Matthew Zurek et al. "Situational Confidence Assistance for Lifelong Shared Autonomy". In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*. 2021, pp. 2783–2789. DOI: 10.1109/ICRA48506.2021.9561839.