# Feature Expansive Reward Learning: Rethinking Human Input

Andreea Bobu*
University of California, Berkeley
abobu@berkeley.edu

Marius Wiggert*
University of California, Berkeley
mariuswiggert@berkeley.edu

Claire Tomlin
University of California, Berkeley
tomlin@berkeley.edu

Anca D. Dragan
University of California, Berkeley
anca@berkeley.edu

## ABSTRACT

When a person is not satisfied with how a robot performs a task, they can intervene to correct it. Reward learning methods enable the robot to adapt its reward function online based on such human input, but they rely on handcrafted features. When the correction cannot be explained by these features, recent work in deep Inverse Reinforcement Learning (IRL) suggests that the robot could ask for task demonstrations and recover a reward defined over the raw state space. Our insight is that rather than *implicitly* learning about the missing feature(s) from demonstrations, the robot should instead ask for data that *explicitly* teaches it about what it is missing. We introduce a new type of human input in which the person guides the robot from states where the feature being taught is highly expressed to states where it is not. We propose an algorithm for learning the feature from the raw state space and integrating it into the reward function. By focusing the human input on the missing feature, our method decreases sample complexity and improves generalization of the learned reward over the above deep IRL baseline. We show this in experiments with a physical 7DOF robot manipulator, as well as in a user study conducted in a simulated environment.

## KEYWORDS

robot learning from human input, human teachers

**ACM Reference Format:**
Andreea Bobu, Marius Wiggert, Claire Tomlin, and Anca D. Dragan. 2021. Feature Expansive Reward Learning: Rethinking Human Input. In *Proceedings of the 2021 ACM/IEEE International Conference on Human-Robot Interaction (HRI '21), March 8–11, 2021, Boulder, CO, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3434073.3444667

## 1 INTRODUCTION

When we deploy robots in human environments, they have to be able to adapt their reward functions to human preferences. For instance in the scenario in Fig. 1, the robot was carrying the cup over the laptop, risking a spill, and the person intervened to correct it. Recent methods interpret such corrections as evidence about the human's desired preference for how to complete the task, enabling the robot to update its reward function online [2, 3, 18].

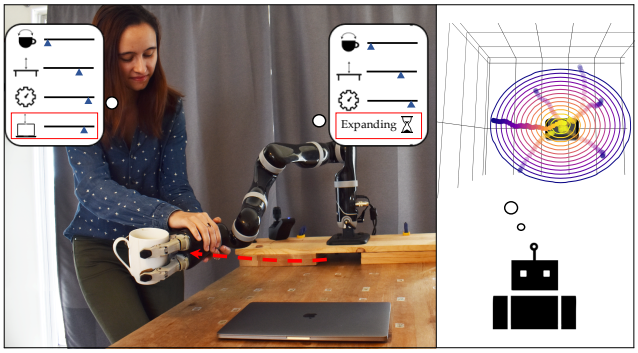


**Figure 1: (Left) After the robot detects that its feature space cannot explain the human's input, the person attempts to teach it the concept of distance from the laptop. (Right) The robot queries the human for feature traces that teach it the missing feature and adapts the reward to account for it.**

Because they have to perform updates online from very little input, these methods resort to representing the reward as a linear function of a small set of hand-engineered *features*. Unfortunately, this puts too much burden on system designers: specifying *a priori* an exhaustive set of *all* the features that end-users might care about is impossible for real-world tasks. While prior work has enabled robots to at least *detect* that the features it has access to cannot explain the human's input [5], it is still unclear how the robot might then construct a feature that can explain it. A natural answer is in deep IRL methods [11, 20, 29], which learn rewards defined directly on the high-dimensional raw state (or observation) space, thereby constructing features automatically. When the robot can't make sense of the human input, it can ask for demonstrations of the task, and learn a reward over not just the known features, but also the raw state space. On the bright side, the learned reward would now be able to explain the demonstrations. On the downside, this may come at the cost of losing generalization when venturing sufficiently far from the demonstrations [14, 24].

In this work, we propose an alternative to relying on demonstrations: we can co-design the learner together with the *type of human*

*Both authors contributed equally to this research.
This research is supported by the Air Force Office of Scientific Research (AFOSR), the Office of Naval Research (ONR-YIP), the DARPA Assured Autonomy Grant, the CONIX Research Center, and the German Academic Exchange Service (DAAD).



*HRI '21, March 8–11, 2021, Boulder, CO, USA.*

ACM ISBN 978-1-4503-8289-2/21/03.
https://doi.org/10.1145/3434073.3444667

*feedback* we ask for. Our insight is that instead of learning about the missing feature(s) *implicitly* through the optimal actions, the robot should ask for data that *explicitly* teaches it what is missing. We introduce a new type of human input, 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 missing feature is highly expressed to states where it is not, in a monotonic fashion.

Looking back at Fig. 1, 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 introduce a reward learning algorithm that harvests the structure inherent to feature traces and uses it to efficiently train a generalizable aspect of the reward: in our example, the distance from the laptop. In experiments on a 7DoF robot arm and in a user study, we find that by taking control of not only the algorithm but also the kind of data it can receive, our method is able to recover more generalizable rewards with much less human input compared to a learning from demonstrations baseline. Although showcased in manipulation, our method should be useful in any scenarios where it is difficult to pre-specify the features the person may care about: in collaborative manufacturing users might care about the rotation of the object handed over, although the robot was only programmed to consider the position, or in autonomous driving passengers may care about how fast to drive through curves.

Finally, we discuss our method's potential implications for the general deep reward learning community. Feature traces enable humans to teach robots about salient aspects of the reward in an intuitive way, making it easier to learn overall rewards. This suggests that taking a divide-and-conquer approach focusing on learning important features separately before learning the reward could benefit IRL generalization and sample complexity. Although more work is needed to teach difficult features with even less supervision, we are excited to have taken a step towards better disambiguating complex reward functions that explain human inputs such as demonstrations in the context of reward learning.

## 2 METHOD

A standard way to teach the robot a reward function over raw state is to provide demonstrations and have the robot learn via deep IRL [11, 29]. Our method introduces a divide-and-conquer alternative, where users can teach the robot explicitly about features that are important, rather than implicitly through full task demonstrations. We assume the robot has access to an initial feature set defined over states $\vec{\phi}(s)$, and is optimizing its current estimate of the reward function, $r_\theta(\vec{\phi}(s))$. Here, $\theta$ is a vector of parameters specifying how the features combine. 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. 2.4). If it can, the robot directly updates its reward function parameters $\theta$, also in line with prior work [2, 23] (Sec. 2.3). But what if it can not? Below, we introduce a method for augmenting the robot's feature space by soliciting further feature-specific human input (Sec. 2.1) and using it to learn a new mapping directly from the raw state space (Sec. 2.2). [1]

### 2.1 Collecting Feature Traces from Humans

A state feature is an arbitrary complex mapping $\phi(s) : \mathbb{R}^d \rightarrow [0, 1]$. To teach a non-linear representation of $\phi$ with little data, we have to balance getting informative inputs and not placing too much burden on the human. As such, we introduce *feature traces* $\xi = s_{0:n}$, a novel type of human input defined as a sequence of $n$ states monotonically decreasing in feature value, i.e. $\phi(s_i) \geq \phi(s_j), \forall i < j$. When learning a feature, the robot queries the human for a set $\Xi$ of $N$ traces. The person gives a trace by simply moving the system from any state $s_0$ to an end state, noisily ensuring monotonicity. Our method, thus, only requires an interface for communicating ordered preferences over states: kinesthetic teaching is useful for household or small industrial robots, while teleoperation and simulation interfaces may be better for larger systems. Because feature learning was triggered by a correction, it is fair to assume that the human knows what aspect of the task they were trying to correct.

To illustrate how a human might offer feature traces, let's turn again to Fig. 1. Here, the person is teaching the robot to keep the mug away from the laptop. The person starts a trace at $s_0$ by placing the end-effector close to 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. 4, with encouraging findings. Further, this limitation can be potentially alleviated using active learning, thereby shifting the burden away from the human to select informative traces, onto the robot to make queries for traces by proposing informative starting states. For instance, the robot could fit an ensemble of functions from traces online, and query for new traces from states where the ensemble disagrees [25].

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 $\xi \in \Xi$ are monotonically decreasing. Since humans are imperfect, we allow users to violate this assumption by modeling them as noisily rational, following the classic Bradley Terry and Luce-Shepard models of preference [6, 21]:

$$P(s > s') = P(\phi(s) > \phi(s')) = \frac{e^{\phi(s)}}{e^{\phi(s)} + e^{\phi(s')}} \quad . \tag{1}$$

Our method also assumes by default that $\phi(s_0) \approx 1$ and $\phi(s_n) \approx 0$, meaning the human starts in a state $s_0$ where the missing feature is highly expressed, then leads the system to a state $s_n$ along decreasing feature values. Since in some situations providing a 1-0 trace is difficult, our algorithm optionally allows the human to give labels $l_0, l_n \in [0, 1]$[2] for the respective feature values.

[1] Prior work proposed tackling this issue by determining which linear feature to add [16], iterative boosting [22], or constructing binary features [8, 19]. We instead focus on unknown features that can be non-binary, non-linear functions of the raw state.
[2] Since providing decimal labels is difficult, the person gives a rating between 0 and 10.

## 2.2 Learning a Feature Function

We represent the missing feature by a neural network $\phi_\psi(s) : \mathbb{R}^d \rightarrow [0, 1]$ . We use the feature traces $\xi$, their inherent monotonicity, and the approximate knowledge about $\phi(s_0)$ and $\phi(s_n)$ to train $\phi_\psi$.

Using Eq. 1, we frame feature learning as a classification problem with a Maximum Likelihood objective over a dataset of tuples $(s, s', y) \in \mathcal{D}$, where $y \in \{0, 0.5, 1\}$ is a label indicating which state has higher feature value. First, due to monotonicity along a feature trace $\xi = (s_0, s_1, \ldots, s_n)$, we have $\phi_\psi(s_i) \geq \phi_\psi(s_j), \forall j > i$, so $y = 1$ if $j > i$ and 0 otherwise. This results in $\binom{(n+1)}{2}$ tuples per trace, which encourage feature values to decrease monotonically along a trace, but they alone don't enforce the same start and end values across traces. Thus, we encode that $\phi(s_0) \approx 1$ and $\phi(s_n) \approx 0$ for all $\xi \in \Xi$ by encouraging indistinguishable feature values at the starts and ends of traces as $(s_0^i, s_0^j, 0.5), (s_n^i, s_n^j, 0.5)\ \forall\ \xi_i, \xi_j \in \Xi,\ i \neq j,\ i > j$. This results in a total dataset of $|\mathcal{D}| = \sum_{i=1}^{N} \binom{(n^i+1)}{2} + 2\binom{N}{2}$ that is already significantly large for a small set of feature traces. The final cross-entropy loss $L(\psi)$ is then given by:

$$-\sum_{(s,s',y)\in\mathcal{D}} y \log(P(s \succ s')) + (1 - y) \log(1 - P(s \succ s')) \quad (2)$$

which expands into the sum of a monotonicity loss for $s \succ s'$ and an indistinguishability loss for $s \sim s'$:

$$-\sum_{s\succ s'} \log \frac{e^{\phi_\psi(s)}}{e^{\phi_\psi(s')} + e^{\phi_\psi(s)}} - \frac{1}{2}\sum_{s\sim s'} \log \frac{e^{\phi_\psi(s)+\phi_\psi(s')}}{(e^{\phi_\psi(s)} + e^{\phi_\psi(s')})^2}. \quad (3)$$

The optional labels $l_0, l_n$ are incorporated as bonus for $s_0, s_n$ as $\phi_\psi(s_0)' = \phi_\psi(s_0) - l_0$ and $\phi_\psi(s_n)' = \phi_\psi(s_n) + (1 - l_n)$ to encourage the labeled feature values to approach the labels. Similar preference learning has been used in imitation and reward learning [9, 17]. The key differences are that the loss in (3) is over feature functions not rewards, and that preferences are provided via feature traces.

## 2.3 Online Reward Update

Once we have a new feature, the robot updates its feature vector $\vec{\phi} \leftarrow (\vec{\phi}, \phi_\psi)$. At this point, the robot goes back to the original human input $a_H$ that previously could not be explained by the old features and uses it to update its estimate of the reward parameters $\theta$. 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 $\tau$ that was optimal under its current reward. Prior work [2] has modeled this as inducing a deformed trajectory $\tau_H$, by propagating the change in configuration to the rest of the trajectory. Further, let $\theta$ define linear weights on the features. Then, the robot updates its estimate $\hat{\theta}$ in the direction of the feature change from $\tau$ to $\tau_H$

$$\hat{\theta}' = \hat{\theta} - \alpha\left(\sum_{s\in\tau_H} \vec{\phi}(s) - \sum_{s\in\tau} \vec{\phi}(s)\right) = \hat{\theta} - \alpha\Big(\Phi(\tau_H) - \Phi(\tau)\Big)\ , \quad (4)$$

where $\Phi$ is the cumulative feature sum along a trajectory. If instead, the human intervened with a full demonstration, work on online learning from demonstrations (Sec. 3.2 in [23]) has derived the same update with $\tau_H$ now the human demonstration. In our implementation, we use corrections and follow [3], which shows that people more easily correct one feature at a time, and only update the $\theta$ 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.

**Algorithm 1:** Feature Expansive Reward Learning (FERL)

**Input:** Features $\vec{\phi} = [\phi_1, \ldots, \phi_f]$, weight $\theta$, confidence threshold $\epsilon$, robot trajectory $\tau$, $N$ number of queries.

```
while executing τ do
    if a_H then
        β̂ ← estimate_confidence(a_H) as in Sec. 2.4.
        if β̂ < ε then
            Ξ = {}
            for i ← 1 to N do
                ξ ← query_feature_trace() as in Sec. 2.1.
                Ξ ← Ξ ∪ ξ.
            φ_new ← learn_feature(Ξ) as in Sec. 2.2.
            φ⃗ ← (φ⃗, φ_new), θ ← (θ, 0.0).
        θ ← update_reward(a_H) as in Sec. 2.3.
        τ ← replan_trajectory(θ).
```

## 2.4 Confidence Estimation

We lastly have to detect that a feature is missing in the first place. Prior work does so by looking at how people's choices are modeled via the Boltzmann noisily-rational decision model:

$$P(\tau_H \mid \theta, \beta) = \frac{e^{\beta R_\theta(\tau_H)}}{\int_{\bar{\tau}_H} e^{\beta R_\theta(\bar{\tau}_H)} d\bar{\tau}_H}\ , \quad (5)$$

where the human picks trajectories proportional to their exponentiated reward [4, 28]. Here, $\beta \in [0, \infty)$ controls how much the robot expects to observe human input consistent with its feature space. Typically, $\beta$ is fixed, recovering the Maximum Entropy IRL [30] observation model. Inspired by work in [5, 12, 13], we instead reinterpret it as a confidence in the robot's features' ability to explain human data. To detect missing features, we estimate $\hat{\beta}$ via a Bayesian belief update $b'(\theta, \beta) \propto P(\tau_H \mid \theta, \beta) b(\theta, \beta)$. If $\hat{\beta}$ is above a threshold $\epsilon$, the robot updates the reward as usual with its current features; if $\hat{\beta} < \epsilon$, the features are insufficient and the robot enters feature learning mode. Algorithm 1 summarizes the full procedure.

# 3 FERL ANALYSIS WITH EXPERT USERS

We first analyze our method (FERL) with real robot data collected from an expert – a person familiar with how the algorithm works – to sanity check its benefits relative to standard reward learning. We will test FERL with non-experts – people not familiar with FERL but are taught to use it – in a user study in Sec. 4.

## 3.1 Feature Learning

Before investigating the benefits of FERL for reward learning, we analyze the quality of the features it can learn. We present results for six different features of varying complexity.

*3.1.1 Experimental Design.* We conduct our experiments on a 7-DoF JACO robotic arm. We investigate six features that arise in the context of personal robotics: a) *table*: distance of the End-Effector

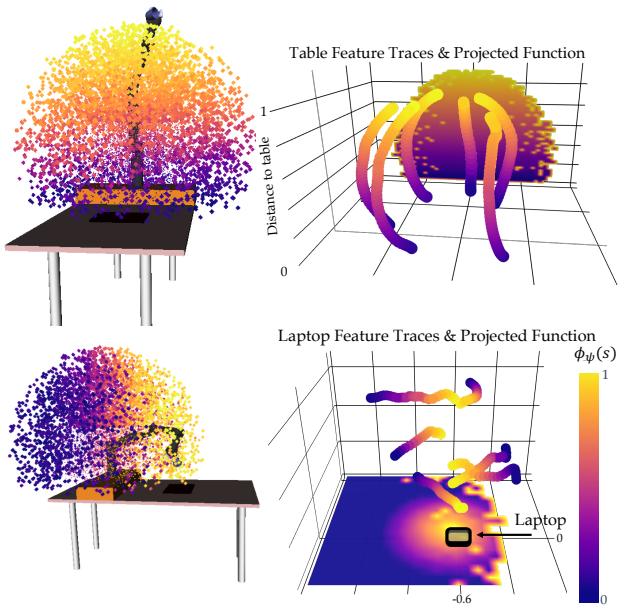


**Figure 2: Visualization of the experimental setup, learned feature values $\phi_\psi(s)$, and training feature traces $\xi$ for *table* (up) and *laptop* (down). We display the feature values $\phi_\psi(s)$ for states $s$ sampled from the reachable set of the 7DoF arm, as well as their projections onto the $yz$ and $xy$ planes.**

(EE) to the table; b) *coffee*: keeping the coffee cup upright; c) *laptop*: 0.3 meter $xy$-plane distance of the EE to a laptop position, to avoid passing over the laptop at any height; d) *test laptop location*: same as *laptop* but the test position differs from the training ones; e) *proxemics*: keeping the EE away from the human, more so when moving in front of them, and less so when moving on their side; f) *between objects*: 0.2 meter $xy$-plane distance of the EE to two objects – the feature penalizes collision with objects, and, to a lesser extent, passing in between the two objects. This feature requires some traces with explicit labels $l_0, l_n$. We approximate all features $\phi_\psi$ by small neural networks (2 layers, 64 units each), and train them on a set $\Xi$ of feature traces using stochastic gradient descent. Our raw state space consists of the 27D $xyz$ positions of all robot joints and objects in the scene, and the rotation matrix of the EE.

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 $\Xi \in \mathcal{F}$ for training. We determine 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. As described in Sec. 2.1, the human teaching the feature 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 provide a diverse dataset.

*Manipulated Variables.* We test feature learning by manipulating the number of traces $N$ the learner gets access to. We would like to see trends in how the quality of the learned features changes with more or less data available.

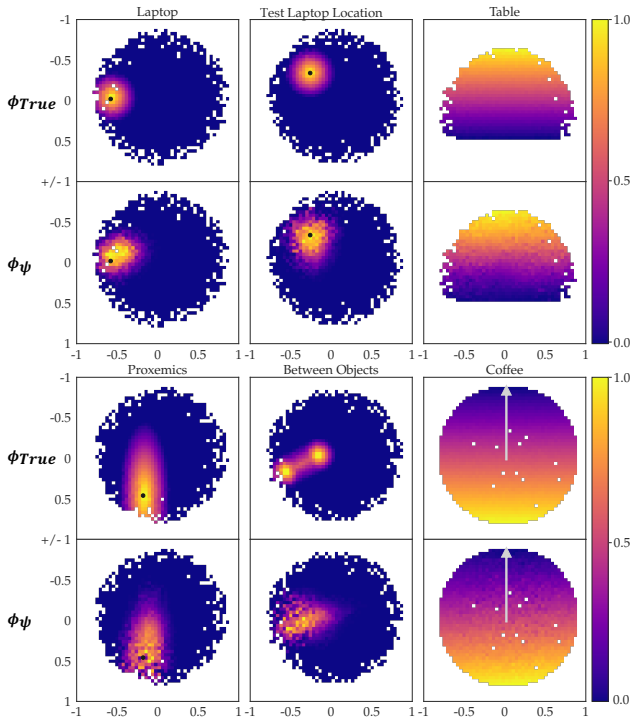


**Figure 3: The plots display the ground truth $\phi_{\text{True}}$ (top rows) and learned feature values $\phi_\psi$ (bottom rows) over $\mathcal{S}_{\text{Test}}$, averaged and projected onto a representative 2D subspace: the $xy$-plane, the $yz$-plane (table), and the $xz$ orientation plane for *coffee* (the arrow represents the cup upright).**

*Dependent Measures.* After training a feature $\phi_\psi$, we measure error compared to the ground truth feature $\phi_{\text{True}}$ 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 $\phi_\psi$. We measure error via the Mean-Squared-Error (MSE), $\text{MSE} = \frac{1}{|\mathcal{S}_{\text{Test}}|} \sum_{\mathcal{S}_{\text{Test}}} ||\phi_\psi(s) - \phi_{\text{True}}(s)||^2$. 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 equals random performance. For each number of feature traces $N$, we run 10 experiments sampling different traces from $\mathcal{F}$, and calculate $\text{MSE}_{\text{norm}}$.

*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.

*3.1.2 Qualitative results.* We first inspect the results qualitatively, for $N = 10$. In Fig. 2 we show the learned *table* and *laptop* features $\phi_\psi$ 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, we illustrate the Ground Truth (GT) feature values

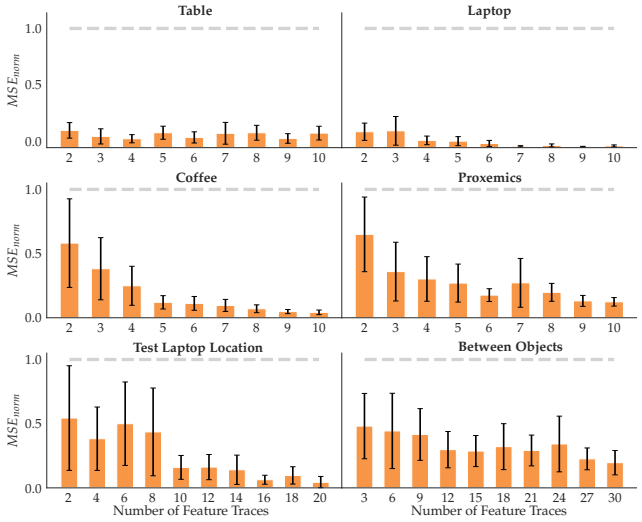


**Figure 4: For each feature, we show the $MSE_{norm}$ mean and standard error across 10 random seeds with an increasing number of traces (orange) compared to random (gray).**

$\phi_{\text{True}}$ and the trained features $\phi_\psi$ 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.

We observe that $\phi_\psi$ resembles $\phi_{\text{True}}$ 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, 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.

*3.1.3 Quantitative analysis.* Fig. 4 displays the means and standard errors across 10 seeds for each feature with data increase. 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 amount of data as a factor and feature ID as a covariate found a significant main effect ($F(8, 526) = 21.1407, p < .0001$). Lastly, in line with 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 the amount of data as a factor, finding a significant main effect ($F(8, 45) = 3.098, p = .0072$). In summary, the qualitative and quantitative results support our hypotheses and suggest that our method requires few traces to reliably learn feature functions $\phi_\psi$ that generalize well to states not seen during training.

## 3.2 Reward Learning

To evaluate FERL for reward learning, we compare it to a deep IRL baseline.

*3.2.1 Experimental Setup.* We run experiments on the robot arm in three settings in which two features are known ($\phi_{\text{coffee}}, \phi_{\text{known}}$) and one feature 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 $\phi_{\text{coffee}}$ with zero weight to evaluate if the methods can learn to ignore irrelevant features. In task 1, $\phi_{\text{laptop}}$ is unknown and the known feature is $\phi_{\text{table}}$; in task 2, $\phi_{\text{table}}$ is unknown and $\phi_{\text{laptop}}$ is known; and in task 3, $\phi_{\text{proxemics}}$ is unknown and $\phi_{\text{table}}$ is known.

*Manipulated Variables.* We manipulated the *learning method* with 2 levels: FERL and an adapted Maximum Entropy Inverse Reinforcement Learning (ME-IRL) baseline [11, 29, 30] learning a deep reward function from demonstrations. We model the ME-IRL reward function $r_\omega$ as a neural network with 2 layers, 128 units each. For a fair comparison, we gave $r_\omega$ access to the known features: once the 27D Euclidean input is mapped to a final neuron, a last layer combines it with the known feature vector.

Also for a fair comparison, we took great care to collect 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. Moreover, FERL and ME-IRL rely on different input types: FERL on feature traces $\xi$ 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 $\Xi$ and demonstrations $\mathcal{D}^*$ such that ME-IRL has more data points: the average number of states per demonstration/trace in our experiments were 61 and 39, respectively.

The gradient of the ME-IRL objective with respect to the reward parameters $\omega$ can be estimated by: $\nabla_\omega \mathcal{L} \approx \frac{1}{|\mathcal{D}^*|} \sum_{\tau \in \mathcal{D}^*} \nabla_\omega R_\omega(\tau) - \frac{1}{|\mathcal{D}^\omega|} \sum_{\tau \in \mathcal{D}^\omega} \nabla_\omega R_\omega(\tau)$ [11, 29]. Here, $R_\omega(\tau) = \sum_{s \in \tau} r_\omega(s)$ is the parametrized reward, $\mathcal{D}^*$ the set of expert demonstrations, and $\mathcal{D}^\omega$ are trajectory samples from the $r_\omega$ induced near optimal policy. We use TrajOpt [27] to obtain the current set of samples $\mathcal{D}^\omega$.

*Dependent Measures.* We compare the two reward learning methods across two metrics commonly used in the IRL literature [7]: 1) *Reward Accuracy*: how close to GT the learned reward is, and 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. For *Reward Accuracy*, we vary the number of traces / demonstrations each learner gets access to, and measure the MSE compared to the GT reward on $\mathcal{S}_{\text{Test}}$, similar to Sec. 3.1. For *Behavior Accuracy*, we train FERL and ME-IRL with a set of 10 traces / demonstrations. We then use TrajOpt [26] to produce optimal trajectories for 100 randomly selected start-goal pairs under the learned rewards. We evaluate the trajectories with the GT reward $r_{\text{true}}$ and divide by the reward of the GT induced trajectory for easy relative comparison.

*Hypotheses.*

**H4:** FERL learns rewards that better generalize to the state space than ME-IRL.

**H5:** FERL performance is less input-sensitive than ME-IRL.

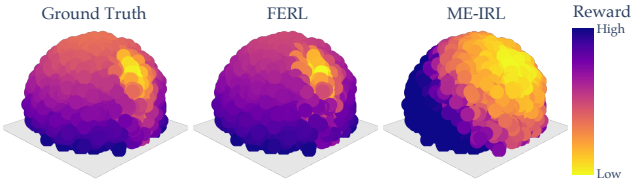


**Figure 5: Visual comparison of the ground truth, FERL, and ME-IRL rewards.**

*3.2.2 Qualitative Comparison.* In Fig. 5, we show the learned FERL and ME-IRL rewards as well as the GT for task 1 evaluated at the test points. As we can see, by first learning the *laptop* feature and then the reward on the extended feature vector, FERL is able to learn a fine-grained reward structure closely resembling GT. Meanwhile, ME-IRL learns some structure capturing where the laptop is, but not enough to result in a good trade-off between the active features.

*3.2.3 Quantitative Analysis.* To compare *Reward Accuracy*, we show in Fig. 6 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 H4. 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. 6 for FERL suggests that our method gets better with more data; in contrast, the same trend is inexistent with ME-IRL. Supporting H5, 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.

Lastly, we looked at *Behavior Accuracy* for the two methods. Fig. 7 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 H4, showing that FERL rewards produce trajectories that are preferred under the GT reward over ME-IRL reward trajectories. An ANOVA using the task as a coviarate reveals a significant main effect for the learning method ($F(1, 596) = 14.9816$, $p = .0001$).

## 4 FERL USER STUDY

In Sec. 3, we analyzed FERL's efficacy with expert data. 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.

### 4.1 Feature Learning

We first collect user feature traces to learn FERL features, and we later test their performance in reward learning in Sec. 4.2.

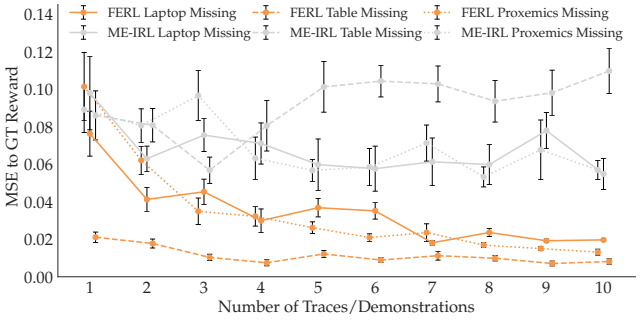


**Figure 6: MSE of FERL and ME-IRL to GT reward.**

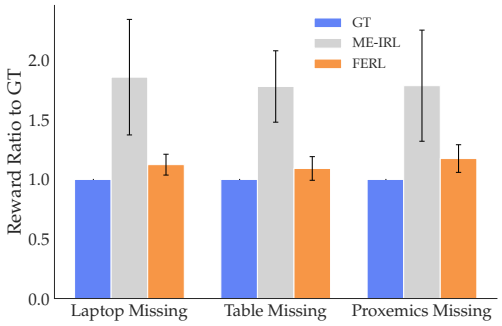


**Figure 7: Induced trajectories' reward ratio.**

*4.1.1 Experimental Design.* Due to COVID, we replicated our setup from Fig. 1 in a pybullet simulator [10] in which users can move a 7 DoF JACO robotic arm using their cursor. Through the interface in Fig. 8 (left), the users can drag the robot to provide feature traces, and use the buttons for recording, saving, and discarding them.

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.

In the second phase, we ask users to teach the robot the three features from Sec. 3.2: *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 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. Additionally, to assess the users' interaction experience we administered the subjective 7-point Likert scale survey from Fig. 9, with some items inspired by NASA-TLX [15]. After they provide the feature

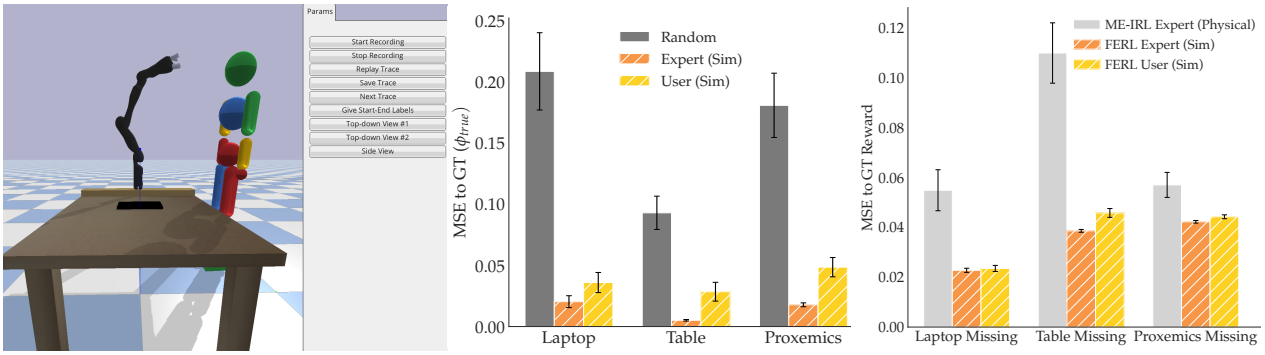


**Figure 8: The user study simulator interface (left). MSE to GT feature comparing expert and non-expert users (middle). MSE to GT reward for ME-IRL with physical demonstrations and FERL with user and expert features learned in simulation (right).**

traces for all 3 tasks, we ask the top eight questions in Fig. 9. The participants then see the 3D visualizations of their feature traces and learned features, and we survey all 11 questions as in Fig. 9 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.* Using our objective and subjective metrics, we test:
**H6:** FERL learns good features even with non-expert user data.
**H7:** 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).

#### 4.1.2 *Analysis.*

*Objective.* Fig. 8 (middle) 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 H6, 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.

*Subjective.* In Fig. 9, 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 are in line with H7, 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

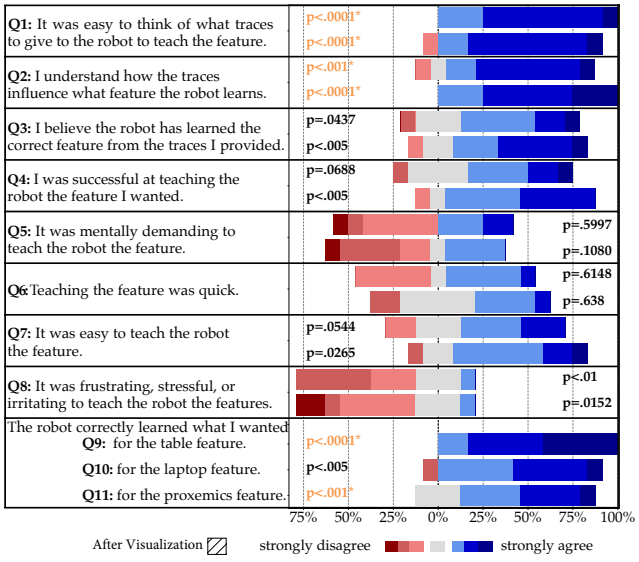


**Figure 9: Questions, answer distributions, and p-values (2-sided t-test against the middle score 4) from the user study. The p-values in orange are significant after adjusted for multiple comparisons using the Bonferroni correction.**

interface, which was necessary because in-person studies are not possible during the COVID pandemic ("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 notice a trend: seeing the result seems to reinforce people's belief that they were effective teachers (Q3, Q4), also noticed in their comments ("Surprised that with limited coverage, it generalized pretty well."). Additionally, in support of H6, 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).

## 4.2 Reward Learning

The objective results in the previous section 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 reward learning.

*4.2.1 Experimental Design.* We take the features learned from the user study, and perform reward learning with them.

*Manipulated Variables.* We manipulate the *learning method*, FERL or ME-IRL, just like in Sec. 3.2. 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. 3.2: the learned reward MSE to the GT reward on $\mathcal{S}_{\text{Test}}$.

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

*4.2.2 Analysis.* Fig. 8 (right) illustrates our findings for the reward comparison. In the figure, we added FERL with expert-taught simulation features for reference: we subsampled sets of 10 from 20 expert traces 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 reward learning: even when given imperfect features, the learned reward is superior to trying to learn everything implicitly from demonstrations.

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

# 5 DISCUSSION

In this work, we proposed FERL, a framework for learning rewards when the initial feature set cannot capture human preferences. We introduced feature traces as human input for learning features from raw state. In experiments and a user study, we showed that FERL outperforms deep reward learning from demonstrations (ME-IRL) in data-efficiency, generalization, and sensitivity to input data.

*Potential 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, e.g. for demonstrations many functions $r_\theta(s)$ induce policies that match the demonstrations' state expectation. The higher dimensional the state $s \in \mathbb{R}^d$, the more human input is needed to disambiguate between those functions sufficiently to find a reward $r_\theta$ which accurately captures human preferences and thereby generalizes to states not seen during training and not just replicates the demonstrations' state expectations as in IRL. We are hopeful that our method of collecting feature traces rather than just demonstrations has potential implications broadly for non-linear (deep) reward learning, as a way to better disambiguate the reward and improve generalization.

While in this paper we focused on adapting a reward online, we also envision our method used as part of a "divide-and-conquer" alternative to IRL: collect feature traces for the salient non-linear criteria of the reward, then use demonstrations to figure out how to combine them. The reason this might 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. These might be the reasons for the result in Fig. 6, 7, 8 (right), where the FERL reward generalized better to new states than the demonstration-only IRL.

*Limitations and Future Work.* Due to the current pandemic, our user study is in a simulated environment instead of in person with the real robot, and most of our users had technical background. While our study still provides evidence that non-expert users can use FERL to teach good features, it is unclear how people without technical background would perform. 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).

Even if people know the missing feature, 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. But for such complex features, it may be more effective to investigate combining feature traces with other types of structured human input.

Lastly, while we show that FERL works reliably in 27D, we want to extend it to higher dimensional state spaces, like images. Approaches that encode these spaces to lower dimensional representations or techniques from causal learning, such as Invariant Risk Minimization [1], could help tackle these challenges.

## REFERENCES


[1] Martín Arjovsky, Léon Bottou, Ishaan Gulrajani, and David Lopez-Paz. 2019. Invariant Risk Minimization. *ArXiv* abs/1907.02893 (2019).

[2] Andrea Bajcsy, Dylan P. Losey, Marcia Kilchenman O'Malley, and Anca D. Dragan. 2017. Learning Robot Objectives from Physical Human Interaction. In *CoRL*.

[3] Andrea Bajcsy, Dylan P. Losey, Marcia K. O'Malley, and Anca D. Dragan. 2018. Learning from Physical Human Corrections, One Feature at a Time. In *Proceedings of the 2018 ACM/IEEE International Conference on Human-Robot Interaction* (Chicago, IL, USA) *(HRI '18)*. ACM, New York, NY, USA, 141–149. https://doi.org/10.1145/3171221.3171267

[4] Chris Baker, Joshua B Tenenbaum, and Rebecca R Saxe. 2007. Goal inference as inverse planning. (01 2007).

[5] A. Bobu, A. Bajcsy, J. F. Fisac, S. Deglurkar, and A. D. Dragan. 2020. Quantifying Hypothesis Space Misspecification in Learning From Human–Robot Demonstrations and Physical Corrections. *IEEE Transactions on Robotics* (2020), 1–20.

[6] Ralph Allan Bradley and Milton E Terry. 1952. Rank analysis of incomplete block designs: I. The method of paired comparisons. *Biometrika* 39, 3/4 (1952), 324–345.

[7] Jaedeug Choi and Kee-Eung Kim. 2011. Inverse reinforcement learning in partially observable environments. *Journal of Machine Learning Research* 12, Mar (2011), 691–730.

[8] Jaedeug Choi and Kee-Eung Kim. 2013. Bayesian nonparametric feature construction for inverse reinforcement learning. In *Twenty-Third International Joint Conference on Artificial Intelligence*.

[9] Paul Christiano, Jan Leike, Tom B. Brown, Miljan Martic, Shane Legg, and Dario Amodei. 2017. Deep reinforcement learning from human preferences. (06 2017).

[10] Erwin Coumans and Yunfei Bai. 2016–2019. PyBullet, a Python module for physics simulation for games, robotics and machine learning. http://pybullet.org.

[11] Chelsea Finn, Sergey Levine, and Pieter Abbeel. 2016. 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* (New York, NY, USA) *(ICML'16)*. JMLR.org, 49–58.

[12] Jaime F Fisac, Andrea Bajcsy, Sylvia L Herbert, David Fridovich-Keil, Steven Wang, Claire J Tomlin, and Anca D Dragan. 2018. Probabilistically Safe Robot Planning with Confidence-Based Human Predictions. *Robotics: Science and Systems (RSS)* (2018).

[13] David Fridovich-Keil, Andrea Bajcsy, Jaime F. Fisac, Sylvia L. Herbert, Steven Wang, Anca D. Dragan, and Claire J. Tomlin. 2019. Confidence-aware motion prediction for real-time collision avoidance. *International Journal of Robotics Research* (2019).

[14] Justin Fu, Katie Luo, and Sergey Levine. 2018. Learning Robust Rewards with Adverserial Inverse Reinforcement Learning. In *International Conference on Learning Representations*. https://openreview.net/forum?id=rkHywl-A-

[15] Sandra G. Hart and Lowell E. Staveland. 1988. Development of NASA-TLX (Task Load Index): Results of Empirical and Theoretical Research. In *Human Mental Workload*, Peter A. Hancock and Najmedin Meshkati (Eds.). Advances in Psychology, Vol. 52. North-Holland, 139 – 183. https://doi.org/10.1016/S0166-4115(08)62386-9

[16] Luis Haug, Sebastian Tschiatschek, and Adish Singla. 2018. Teaching inverse reinforcement learners via features and demonstrations. In *Advances in Neural Information Processing Systems*. 8464–8473.

[17] Borja Ibarz, Jan Leike, Tobias Pohlen, Geoffrey Irving, Shane Legg, and Dario Amodei. 2018. Reward learning from human preferences and demonstrations in Atari. In *Advances in Neural Information Processing Systems*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett (Eds.), Vol. 31. Curran Associates, Inc., 8011–8023. https://proceedings.neurips.cc/paper/2018/file/8cbe9ce23f42628c98f80fa0fac8b19a-Paper.pdf

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

[19] Sergey Levine, Zoran Popovic, and Vladlen Koltun. 2010. Feature construction for inverse reinforcement learning. In *Advances in Neural Information Processing Systems*. 1342–1350.

[20] Sergey Levine, Zoran Popovic, and Vladlen Koltun. 2011. Nonlinear inverse reinforcement learning with gaussian processes. In *Advances in Neural Information Processing Systems*. 19–27.

[21] R. Duncan Luce. 1959. *Individual choice behavior*. John Wiley, Oxford, England. xii, 153–xii, 153 pages.

[22] Nathan Ratliff, David M Bradley, Joel Chestnutt, and J A Bagnell. 2007. Boosting structured prediction for imitation learning. In *Advances in Neural Information Processing Systems*. 1153–1160.

[23] Nathan D. Ratliff, J. Andrew Bagnell, and Martin A. Zinkevich. 2006. Maximum Margin Planning. In *Proceedings of the 23rd International Conference on Machine Learning* (Pittsburgh, Pennsylvania, USA) *(ICML '06)*. Association for Computing Machinery, New York, NY, USA, 729–736. https://doi.org/10.1145/1143844.1143936

[24] Siddharth Reddy, Anca D. Dragan, and Sergey Levine. 2020. 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. https://openreview.net/forum?id=S1xKd24twB

[25] Siddharth Reddy, Anca D. Dragan, Sergey Levine, Shane Legg, and Jan Leike. 2019. Learning Human Objectives by Evaluating Hypothetical Behavior. arXiv:1912.05652 [cs.CY]

[26] John Schulman, Jonathan Ho, Alex Lee, Ibrahim Awwal, Henry Bradlow, and Pieter Abbeel. [n.d.]. Finding Locally Optimal, Collision-Free Trajectories with Sequential Convex Optimization.

[27] John Schulman, Jonathan Ho, Alex X Lee, Ibrahim Awwal, Henry Bradlow, and Pieter Abbeel. 2013. Finding Locally Optimal, Collision-Free Trajectories with Sequential Convex Optimization.. In *Robotics: science and systems*, Vol. 9. Citeseer, 1–10.

[28] John Von Neumann and Oskar Morgenstern. 1945. *Theory of games and economic behavior*. Princeton University Press Princeton, NJ.

[29] M. Wulfmeier, D. Z. Wang, and I. Posner. 2016. Watch this: Scalable cost-function learning for path planning in urban environments. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2089–2095.

[30] Brian D. Ziebart, Andrew Maas, J. Andrew Bagnell, and Anind K. Dey. 2008. Maximum Entropy Inverse Reinforcement Learning. In *Proceedings of the 23rd National Conference on Artificial Intelligence - Volume 3* (Chicago, Illinois) *(AAAI'08)*. AAAI Press, 1433–1438. http://dl.acm.org/citation.cfm?id=1620270.1620297