Active Preference-Based Gaussian Process Regression for Reward Learning

Erdem Bıyık*
Electrical Engineering
Stanford University
ebiyik@stanford.edu

Nicolas Huynh*
Applied Mathematics
École Polytechnique
nicolas.huynh@polytechnique.edu

Mykel J. Kochenderfer
Aeronautics & Astronautics
Stanford University
mykel@stanford.edu

Dorsa Sadigh
Computer Science
Stanford University
dorsa@cs.stanford.edu

***Abstract*— Designing reward functions is a challenging problem in AI and robotics. Humans usually have a difficult time directly specifying all the desirable behaviors that a robot needs to optimize. One common approach is to learn reward functions from collected expert demonstrations. However, learning reward functions from demonstrations introduces many challenges: some methods require highly structured models, e.g. reward functions that are linear in some predefined set of features, while others adopt less structured reward functions that on the other hand require tremendous amount of data. In addition, humans tend to have a difficult time providing demonstrations on robots with high degrees of freedom, or even quantifying reward values for given demonstrations. To address these challenges, we present a preference-based learning approach, where as an alternative, the human feedback is only in the form of comparisons between trajectories. Furthermore, we do not assume highly constrained structures on the reward function. Instead, we model the reward function using a Gaussian Process (GP) and propose a mathematical formulation to *actively* find a GP using only human preferences. Our approach enables us to tackle both inflexibility and data-inefficiency problems within a preference-based learning framework. Our results in simulations and a user study suggest that our approach can efficiently learn expressive reward functions for robotics tasks.**

I. Introduction

Planning for robots that can act in a diverse set of environments based on human preferences can be quite challenging. It is generally infeasible for human designers to directly program the desired behavior for the full spectrum of possible situations. Hence, roboticists often use machine learning in at least part of their design to discover human preferences. One approach is to directly learn a robot policy using expert demonstrations [1]–[4]. However, in many interactive settings, we are interested in more generally learning a reward function that represents how a robot should act or interact in the world.

Reward functions are powerful tools for specifying desirable robot behaviors, e.g. how to act safely, or what styles or goals the robot needs to follow. Unfortunately, specifying reward functions is not an easy task for human designers [5]–[7]. Our goal in this work is to develop a *data-efficient* method that can learn *expressive* reward functions.

Prior work has considered using a sequence of pairwise comparisons between trajectories as a successful technique to learn reward functions [8]–[15]. For example, as shown in Fig. 1, the robot can demonstrate the blue and green trajectories, ξ_A and ξ_B, and ask the human designer to compare the two. Preference-based reward learning can then leverage a sequence of pairwise comparisons to accurately estimate a reward function.

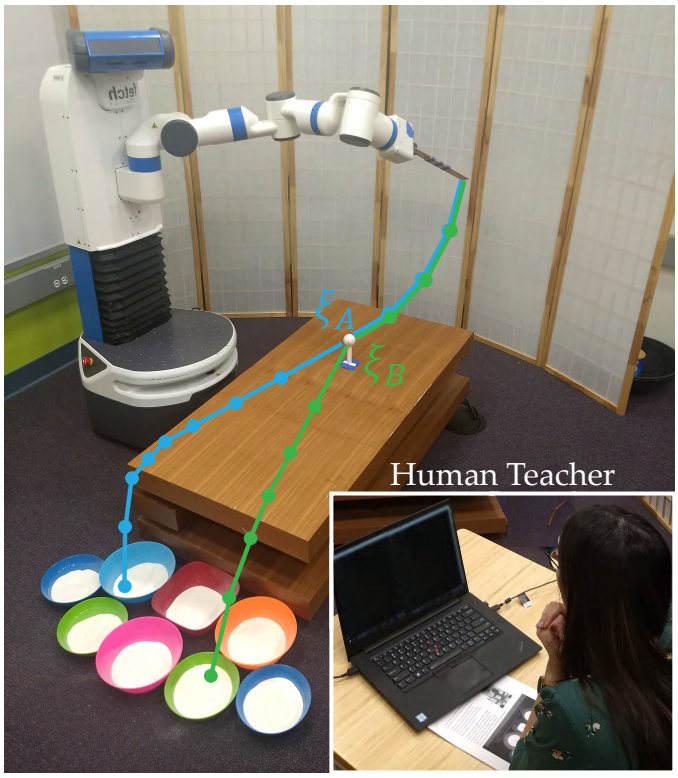


Fig. 1. The user is trying to teach the robot how to play a variant of mini-golf, where the reward differs among eight targets. In preference-based learning, instead of trying to design a reward function by hand or controlling the robot to provide demonstrations, the user simply compares two demonstrated trajectories on the robot. Here, ξ_A and ξ_B demonstrate two trajectories that correspond to hitting the ball towards the blue or green targets.

However, preference-based learning techniques are in general not very data-efficient, as each pairwise comparison only provides 1 bit of information, i.e., if ξ_A is preferred over ξ_B or vice versa. Therefore, active learning is commonly employed in this framework to find the most informative or diverse sequence of questions for efficiently converging to the underlying reward function [12]–[14], [16]–[23].

Unfortunately, most prior active reward learning works rely

*First two authors contributed equally and are listed in alphabetical order.

on a strong assumption about the structure of the reward function, i.e., *the reward function is a linear combination of a set of hand-coded features*. While this assumption is commonly needed for active learning to scale, it is very limiting because linear reward functions are not sufficiently expressive. For example, a linear reward would require several features for the human teacher to be able to specify every reward configuration of targets in the task demonstrated in Fig. 1, i.e., how targets compare to each other. The features to this task could be, for example, distances to each and every target. On the other hand, if the reward model was nonlinear, one can capture all possible configurations with only two features: speed for how far the ball will be thrown, and angle for which direction to shoot. While neural networks or kernel functions can provide this flexibility, these techniques considerably increase the number of parameters needed, which prohibits (or renders useless the advantage of) active learning algorithms.

> Our key insight is to model the reward function using a Gaussian Process (GP) [44]. GPs are non-parametric models that can capture nonlinearities, allowing us to actively and efficiently learn highly expressive reward functions.

In this work, we propose a mathematical framework for *actively* fitting a GP using only pairwise comparisons between two trajectories, which we call *preference* data. Leveraging GPs, instead of linear models with hand-designed features, improves the expressiveness of reward functions by incorporating nontrivial nonlinearities. Besides, our active query generation method enables us to still get the benefits of active learning.

We make two main contributions in this work:

- We propose a *data-efficient* mathematical framework for actively regressing a GP with preference data to learn *expressive* reward functions from humans.
- We demonstrate the performance of our framework through simulated environments and user studies on a manipulator robot playing a variant of mini-golf based on different human preferences. Our results show our approach can be used for reward learning in complex and interesting settings and is more data-efficient than various other baselines.

II. Related Work

In this section, we will discuss the prior work focused on learning reward functions from demonstration, or other sources of data, as well as related work in Gaussian process regression and its relation to our work.

Learning reward functions from demonstrations. Prior work has studied learning reward functions from collected expert demonstrations. This is commonly referred to as inverse reinforcement learning (IRL), where we assume human demonstrations are based on them approximately optimizing a reward function [24]–[28]. The learned reward function can then be used by a robot to optimize its actions in the broad range of environments.

While IRL has shown promising results in a variety of domains, robots, especially manipulators with high degrees of freedom, are often too difficult to manually operate [29]–[33]. Moreover, recent studies in autonomous driving, where the high degrees of freedom of a robot is not an issue, suggest that people do not prefer an autonomous car to follow their own demonstrations and instead prefer a different behavior that tends to be more timid [34]. These indicate that one needs to go beyond human demonstrations to properly capture the preferred reward function.

In our framework, instead of relying on human demonstrations, we learn the reward functions through the preference queries. Therefore, our method does not require experts who can control the system in the desirable way.

Learning reward functions from other sources of data. In addition to demonstrations and physical corrections [35], [36], where the robot attempts to learn the reward function through physical human interference, learning from rankings [10] is another popular approach. A particular case of this is when the rankings are only pairwise comparisons, which we call preference queries. Previous works have investigated the use of preference queries for reward learning. Sadigh *et al.* [12] proposed an acquisition function to actively generate the queries. Further studies extended this approach to batch-active methods [16], [17], rankings instead of pairwise comparisons [22], general Markov Decision Process (MDP) settings [21], and settings that integrate expert demonstrations with preference queries [20]. The reward function these prior works assume is linear with respect to some hand-coded features. This assumption limits the model flexibility and requires very careful feature design. Basu *et al.* [19] explored modeling a multi-modal reward function, but the reward was still linear in each mode. Furthermore, they focused only on bi-modal distributions. Scalability to more modes remains an issue.

In this work, we do not make the linearity assumption and instead model the reward using a GP. Our results show this significantly improves the expressive power of the learned reward function, and the method is still very data-efficient.

Gaussian process regression. On the machine learning side, González *et al.* [37] and Chu *et al.* [38] proposed methods for preference-based Bayesian optimization and GP regression, respectively, but they were not active. Furthermore, [37] required to regress a GP over $2d$-dimensions to model a d-dimensional function, which causes a computational burden. More relevantly, Houlsby *et al.* [39] presented an active method for preference-based GP regression. However, similar to [37], the regression was over a $2d$-dimensional space where the learned model predicts the outcome of a comparison rather than outputting a reward value. Jensen *et al.* [40] showed how to update a GP with preference data, but the active query generation was not an interest. Kapoor *et al.* [41] developed an active learning approach for classification with GPs. This is a specific case of our problem, as the labels in classification are consistent, i.e., the labels assigned to the samples in the dataset, even if they are incorrect, do not change during training. In our case, the

user can respond to the same preference query inconsistently depending on their noise model. Houlsby *et al.* [42] and Daniel *et al.* [43] proposed active GP fitting methods for classification and reward learning, respectively. While the latter focused on robotics tasks, they were not preference-based. Hence, they may be infeasible in many applications as it is difficult for humans to assign actual reward values.

In this work, we propose an active query generation method for preference-based GP regression. While being data-efficient, this method also does not require the humans to assign actual reward values to the trajectories for fitting the GP.

III. Problem Formulation

We model the environment the robot is going to operate in as a finite-horizon deterministic MDP. We use $s_t \in \mathcal{S}$ to denote the state and $a_t \in \mathcal{A}$ for the action (control inputs) at time t. A trajectory $\xi \in \Xi$ within this MDP is a sequence that consists of the initial position and the actions: $\xi = (s_0, a_0, a_1, \ldots, a_T)$, where T is the finite time horizon.

We assume a reward function over trajectories, $R : \Xi \to \mathbb{R}$, that encodes the human user's preferences about how they want the system to operate.

We assume the reward function R can be formulated as $R(\xi) = f(\Psi(\xi))$, where $\Psi : \Xi \to \mathbb{R}^d$ defines a set of trajectory features, e.g. average speed and minimum distance to the closest car in a driving trajectory. However, we emphasize that this formulation of R enables a more general form of functions that does not require strong assumptions – such as linearity in the features – which is commonly put in place when learning reward functions. We use a GP to model f, which allows us to avoid strong assumptions about the form of f.[1]

Our goal is to learn this more general form of reward functions using preference data in the form of pairwise comparisons. The robot will demonstrate a query Q consisting of two trajectories, ξ_A and ξ_B as shown in Fig. 1 with blue and green curves, to the user, and will ask which trajectory they prefer. The user will respond to this query based on their preferences. The user's response provides useful information about the underlying preference reward function R. Of course, we cannot assume human responses are perfect for every query, so we model the noise in their responses using the commonly adopted probit model, which assumes humans make a binary decision between the two trajectories noisily based on the cumulative distribution function (cdf) of the difference between the two reward values:

$$P(q = \xi_A \mid Q = \{\xi_A, \xi_B\}) = P\left(R(\xi_A) - R(\xi_B) > v\right),$$

where $q \in Q$ denotes the user's choice, and $v \sim \mathcal{N}(0, 2\sigma^2)$ for some standard deviation $\sigma\sqrt{2}$. Therefore, equivalently:

$$P(q = \xi_A \mid Q = \{\xi_A, \xi_B\}) = \Phi\left(\frac{R(\xi_A) - R(\xi_B)}{\sqrt{2}\sigma}\right), \quad (1)$$

where Φ is the cumulative distribution function of the standard normal.

[1]Due to computation reasons, we assume d is small. Compared to previous works which assume R to be linear in features, this is a very mild assumption.

Having defined the problem setting, we are now ready to present our method for learning data-efficient and expressive reward functions using GPs.

IV. Methods

In this section, we first give some background information about Gaussian Processes. We then introduce preference-based GP regression, where we show how to update a GP with the results of pairwise comparisons. Finally, we present our approach to active preference query generation, where we discuss how to find the most informative query that accelerates the learning.[2] To simplify the notation, we replace $\Psi(\xi)$ with Ψ, with superscripts and subscripts when needed.

A. Gaussian Processes

We start by introducing the necessary background on GPs for our work. We refer the readers to [44] for other uses of GPs in machine learning.

Suppose we are given a dataset $\mathbf{\Psi} = (\Psi_i)_{i=1}^N$, where $\Psi_i \in \mathbb{R}^d$. We employ a probabilistic point of view for f by modeling it using a GP, which enables us to model nonlinearities and uncertainties well without introducing parameters. We have:

$$P(\mathbf{f} \mid \boldsymbol{\mu}, \mathbf{K}) = \frac{\exp\left(-\frac{1}{2}(\mathbf{f} - \boldsymbol{\mu})^\top \mathbf{K}^{-1}(\mathbf{f} - \boldsymbol{\mu})\right)}{(2\pi)^{N/2}|\mathbf{K}|^{1/2}}, \quad (2)$$

where $\mathbf{f} = (f(\Psi_i))_{i=1}^N$, $\boldsymbol{\mu}$ and $\mathbf{K}$ are the mean vector and the covariance matrix of the GP distribution for the N items in the dataset. Put it in another way, $\mathbf{f}$ follows a multivariate distribution. The covariance matrix depends on the used kernel. In this work, we use a variant of radial basis function (RBF) kernel with hyperparameter θ:

$$k(\Psi_i, \Psi_j) = \exp\left(-\theta\|\Psi_i - \Psi_j\|_2^2\right) - \bar{k}(\Psi_i, \Psi_j),$$
$$\bar{k}(\Psi_i, \Psi_j) = \exp\left(-\theta\|\Psi_i - \bar{\Psi}\|_2^2 - \theta\|\Psi_j - \bar{\Psi}\|_2^2\right),$$

where $\bar{\Psi} \in \mathbb{R}^d$ is an arbitrary point for which we assume $f(\bar{\Psi}) = 0$. This is important because the query responses only depend on the relative difference between the two reward function values at the trajectories, i.e., $f(\Psi) + c$ for any $c \in \mathbb{R}$ would have the same likelihood for a dataset as $f(\Psi)$. By setting $f(\bar{\Psi}) = 0$ for some arbitrary $\bar{\Psi} \in \mathbb{R}^d$, we dissolve this ambiguity. It does not introduce an assumption because for any function f' and for any point $\bar{\Psi}$, one can define $f(\Psi) = f'(\Psi) - f'(\bar{\Psi})$ without loss of generality—both f' and f will encode the same preferences. Finally, this variant of the RBF kernel is still positive semi-definite, because it is equivalent to the covariance kernel of a GP which is initialized with an initial data point and a standard RBF kernel prior.

B. Preference-based GP Regression

In preference-based learning, we have a dataset $\mathbf{\Psi} = ((\Psi_i^{(1)}, \Psi_i^{(2)}))_{i=1}^N$, consisting of pairs of trajectories $\Psi_i^{(1)}, \Psi_i^{(2)} \in \mathbb{R}^d$, and user responses $\mathbf{q} = (q_i)_{i=1}^N$, where $q_i \in \{0, 1\}$ indicates whether the user preferred $\Psi_i^{(1)}$ or $\Psi_i^{(2)}$. Accordingly, $\mathbf{K}$ is now a $2N \times 2N$ matrix, whose rows and

[2]We make our Python code for active query generation publicly available at https://github.com/Stanford-ILIAD/active-preference-based-gpr.

columns correspond to $\Psi_i^j, \forall i \in \{1,\dots,N\}, \forall j \in \{1,2\}$. Similarly, $\boldsymbol{\mu}$ is a $2N$-vector. Using a Bayesian approach to update the GP with new preference data (Ψ, q) gives:

$$\begin{aligned} P(f \mid \boldsymbol{\mu}, \mathbf{K}, \Psi, q) &\propto P(q \mid f, \boldsymbol{\mu}, \mathbf{K}, \Psi) P(f \mid \boldsymbol{\mu}, \mathbf{K}, \Psi) \\ &= P(q \mid f, \Psi) P(f \mid \boldsymbol{\mu}, \mathbf{K}). \end{aligned} \tag{3}$$

Here, the first term is just the probabilistic human response model given in Eqn. (1), and the second term is Eqn. (2). However, this posterior does not follow a GP distribution similar to Eqn. (2), and becomes analytically intractable [40].

Prior works have shown it is possible to perform some approximation such that the posterior is another GP [40], [44]. The most common approximation techniques are

- Laplace approximation, where the idea is to model the posterior as a GP such that the mode of the likelihood is treated as the posterior mean, and a second-order Taylor approximation around the maximum of the log-likelihood gives the posterior covariance. This technique is computationally very fast.
- Expectation Propagation (EP), where the idea is to approximate each factor of the product by a Gaussian. EP is an iterative method that processes each factor iteratively to update the distribution to minimize its KL-divergence with the true posterior. While it is more accurate than Laplace approximation, it is slower in practice [45].

In this paper, we use the former for its computational efficiency. Hence, we now show how to compute the quantities for Laplace approximation, i.e., posterior mean and covariance.

Finding the posterior mean. We use the mode of the posterior as an approximation to the posterior mean:

$$\arg\max_{\mathbf{f}} \left(\log\left(p(\mathbf{q} \mid \boldsymbol{\Psi}, \mathbf{f})\right) + \log\left(P(\mathbf{f} \mid \boldsymbol{\Psi})\right)\right) \tag{4}$$

Because the preference data are conditionally independent, the first term can be written as:

$$\begin{aligned} \log\left(P(\mathbf{q} \mid \boldsymbol{\Psi}, \mathbf{f})\right) &= \sum_{i=1}^{N} \log P(q_i \mid \Psi_i, \mathbf{f}) \\ &= \sum_{i=1}^{N} \log \Phi\left(\frac{f(\Psi_i^{(1)}) - f(\Psi_i^{(2)})}{\sqrt{2}\sigma}\right) \end{aligned}$$

due to Eqn. (1). Adopting a zero-mean prior for f, we can write the second term of the optimization (4) as:

$$\log\left(P(\mathbf{f} \mid \boldsymbol{\Psi})\right) = -\frac{1}{2}\log|\mathbf{K}| - N\log 2\pi - \frac{1}{2}\mathbf{f}^\top \mathbf{K}^{-1}\mathbf{f}$$

Armed with these two expressions, we can now optimize the log-likelihood and thus find the mode of it to approximate the posterior mean.

Finding the posterior covariance matrix. Following [44], and omitting the derivation details for brevity, the posterior covariance matrix is $(\mathbf{K}^{-1} + \mathbf{W})^{-1}$ where $\mathbf{W}$ is the negative Hessian of the log-likelihood:

$$W_{ij} = -\frac{\partial^2 \log\left(P(\mathbf{q} \mid \boldsymbol{\Psi}, \mathbf{f})\right)}{\partial f^{(i)} \partial f^{(j)}}.$$

Now, we know how to approximate the posterior mean and the posterior covariance for the Laplace approximation of Eqn. (3). This allows us to model and update the reward with preference data using a GP.

We also want to perform inference from this approximated GP. Inference is not only useful for active query generation, but it also enables us to compute the reward expectations and variances given a trajectory.

Inference. Given two points $\Psi_*^{(1)}, \Psi_*^{(2)} \in \mathbb{R}^d$, we want to be able to compute the expected mean rewards $\boldsymbol{\mu}_*$ and also the covariance matrix between those two points $\mathbf{K}_*$, both of which will be useful for active query generation. These are given by:

$$\mu_* = \mathbb{E}\left[\mathbf{f}_* \mid \boldsymbol{\Psi}, \mathbf{q}, \Psi_*^{(1)}, \Psi_*^{(2)}\right] = k_*^\top \mathbf{K}^{-1}\mathbf{f}, \tag{5}$$

where $k_{*ij} = k(\Psi_*^{(i)}, \Psi_j)$ is a $2 \times 2N$ matrix, and

$$\mathbf{K}_* = \boldsymbol{K_0} - k_*\left(\boldsymbol{I}_{2N} + \mathbf{W}\mathbf{K}\right)^{-1}\mathbf{W}k_*^\top, \tag{6}$$

where $K_{0ij} = k\left(\Psi_*^{(i)}, \Psi_*^{(j)}\right)$ is a 2×2 matrix, $\boldsymbol{I}_{2N}$ is the $2N \times 2N$ identity matrix.

Equipped with all these tools which enable us to approximate the posterior distribution with a GP and perform inference over it, we are now ready to present our contributions on the active query generation.

C. Active Query Synthesis for Reward Learning

While we now know how to learn the reward function f using only pairwise comparisons, this endeavor can require tremendous amount of data, because each query will give at most 1 bit of information. Furthermore, we can expect a decreasing trend in the information gain as we learn the reward function. Therefore, it is important to select the queries for the human such that each query gives as much information as possible. For linear reward models, previous work has shown that this can be done by maximizing the mutual information, which also makes the queries easy for the user [18]. Extending this formulation to the reward functions modeled with a GP is not trivial, because one needs to sample from the GP many times for each trajectory, whereas a linear reward form allows the reward prediction after sampling the linear weight terms only once.

Hence, for the active query generation, our goal is to perform information gain maximization with GPs.

Problem 1. *Formally, we want to solve the following problem:*

$$\Psi_*^{(1)}, \Psi_*^{(2)} = \arg\max_{\Psi^{(1)}, \Psi^{(2)}} I(f; q \mid \Psi, \boldsymbol{\Psi}, \mathbf{q}),$$

where I is the mutual information and q is the response to the query $\Psi = (\Psi^{(1)}, \Psi^{(2)})$. This optimization is equivalent to

$$\arg\max_{\Psi^{(1)}, \Psi^{(2)}} \left(H(q \mid \Psi, \boldsymbol{\Psi}, \mathbf{q}) - \mathbb{E}_{f \sim P(f \mid \boldsymbol{\Psi}, \mathbf{q})}\left[H(q \mid \Psi, f)\right]\right), \tag{7}$$

where H is the information entropy.

This optimization can be interpreted as follows: On one hand, maximizing the first entropy term $H(q \mid \Psi, \boldsymbol{\Psi}, \mathbf{q})$ encourages fast convergence by maximizing the uncertainty of the outcome of every query for the learned GP model. On the

other hand, minimizing the second entropy term $H(q \mid \Psi, f)$ encourages the ease of responding to the queries by the user meaning the user should be certain about their choices.

We defer the full derivation of (7) to the appendix, but here we give an easy-to-implement formulation of the optimization. Denoting the posterior mean of $f(\Psi^{(i)})$, which is obtained using Eqn. (5), with $\mu^{(i)}$, the objective function can be written as

$$h\left(\Phi\left(\frac{\mu^{(1)}-\mu^{(2)}}{\sqrt{2\sigma^2+g(\Psi^{(1)},\Psi^{(2)})}}\right)\right)-m\left(\Psi\right) \tag{8}$$

where

$$\begin{aligned} g(\Psi^{(1)},\Psi^{(2)}) =& \operatorname{Var}\left(f(\Psi^{(1)})\right)+\operatorname{Var}\left(f(\Psi^{(2)})\right) \\ &-2\operatorname{Cov}\left(f(\Psi^{(1)}),f(\Psi^{(2)})\right), \end{aligned}$$

whose terms can be computed using Eqn. (6); h is the binary entropy function, i.e.,

$$h(p)=-p\log_2(p)-(1-p)\log_2(1-p),$$

and

$$m\left(\Psi\right)=\frac{\sqrt{\pi\ln(2)\sigma^2}\exp\left(-\frac{(\mu^{(1)}-\mu^{(2)})^2}{\pi\ln(2)\sigma^2+2g(\Psi^{(1)},\Psi^{(2)})}\right)}{\sqrt{\pi\ln(2)\sigma^2+2g(\Psi^{(1)},\Psi^{(2)})}}.$$

Synthesizing queries that maximize this objective will give us very informative data points for preference-based GP regression and improve data-efficiency.

Previously, Biyik *et al.* [18] have shown for the linear reward models that using an information gain based formulation accelerates the learning whereas volume removal based methods (such as [12]) rely on local optima and can produce trivial queries that compare the exact same trajectory and so gives no information. In the following, we show our formulation also does not suffer from this trivial query problem.

Remark 1. *The trivial query* $\Psi=\{\Psi^{(A)},\Psi^{(A)}\}$ *does not maximize our acquisition function given in* (8)*, and is in fact a global minimizer.*

Proof: For the query $\Psi=\{\Psi^{(A)},\Psi^{(A)}\}$, we rewrite (8) as

$$h\left(\Phi\left(\frac{\mu^{(A)}-\mu^{(A)}}{\sqrt{2\sigma^2+g(\Psi^{(A)},\Psi^{(A)})}}\right)\right)-m\left(\Psi\right)=1-m(\Psi)$$

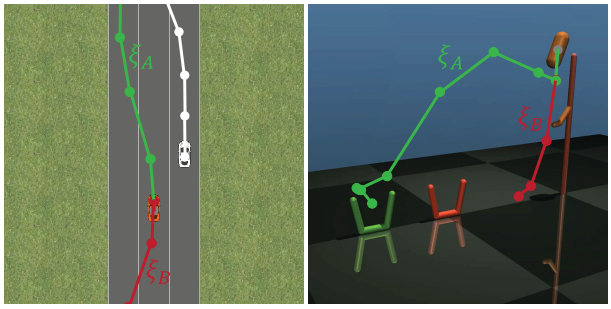
where $\operatorname{Var}\left(f(\Psi^{(A)})\right)=\operatorname{Cov}\left(f(\Psi^{(A)}),f(\Psi^{(A)})\right)$, and so $g(\Psi^{(A)},\Psi^{(A)})=0$, and

$$m\left(\Psi\right)=\frac{\sqrt{\pi\ln(2)\sigma^2}\exp\left(-\frac{(\mu^{(A)}-\mu^{(A)})^2}{\pi\ln(2)\sigma^2+2g(\Psi^{(A)},\Psi^{(A)})}\right)}{\sqrt{\pi\ln(2)\sigma^2+2g(\Psi^{(A)},\Psi^{(A)})}}=1$$

which makes the objective value 0. Since the information gain has to be nonnegative, this completes the proof that the trivial query is a global minimizer of the objective. ■

V. Simulation Experiments

In this section, we present our experiments in two simulation domains to demonstrate how (i) GP rewards improve expressiveness over linear reward functions, and (ii) active query generation improves data-efficiency over random querying.



Driver Tosser

Fig. 2. Sample trajectories are shown for the two simulation environments. In *Driver*, another car is cutting in front of the ego vehicle. In *Tosser*, the robot must hit the dropping capsule such that it will fall into one of the baskets.

A. Simulation Environments

To validate our framework on robotics tasks, we used two simulation environments: a 2D *Driver* simulation [46] and a MuJoCo [47] environment to simulate a *Tosser* robot that tries to throw an object into a basket [16]. We show visuals from these environments with sample trajectories in Fig. 2. For example in *Driver*, the user is asked whether they would move forward or backward in the given scenario. While the users would have a common response to this query, some questions may differ among the users. For instance in Tosser, the query asks the user whether to throw the ball into the green basket or to drop it instead. Depending on the users' preferences about the green basket, different users may have different responses.

In these two environments, we use the following simple features for the function Ψ similar to [16]:

- *Driver*: Distance to the other car, speed, heading angle, distance to the closest lane center.
- *Tosser*: The maximum horizontal range, and the number of capsule flips.

In contrast to what the previous work reported, here we do not need to fine-tune the feature parameters to learn the reward functions because GPs can effectively capture nonlinearities.

Simulated Human Model. We simulated human responses with an underlying true reward function f with some Gaussian noise, in accordance with Eqn. (1). We modeled f as either a degree-of-two polynomial or a linear function. In both cases, we selected the parameters of f as i.i.d. random samples from the standard normal distribution. We repeated each simulation experiment 5 times with varying underlying true reward functions.

B. Baselines

For our analyses, we compared three methods:

- RandomGP: The reward is modeled using a Gaussian Process. The two distinct trajectories selected in each training query are sampled from a training dataset uniformly at random.
- ActiveLinear: The reward is modeled as a linear combination of features, and the active query generation method of [18] selects the most informative comparison queries at every step of training.

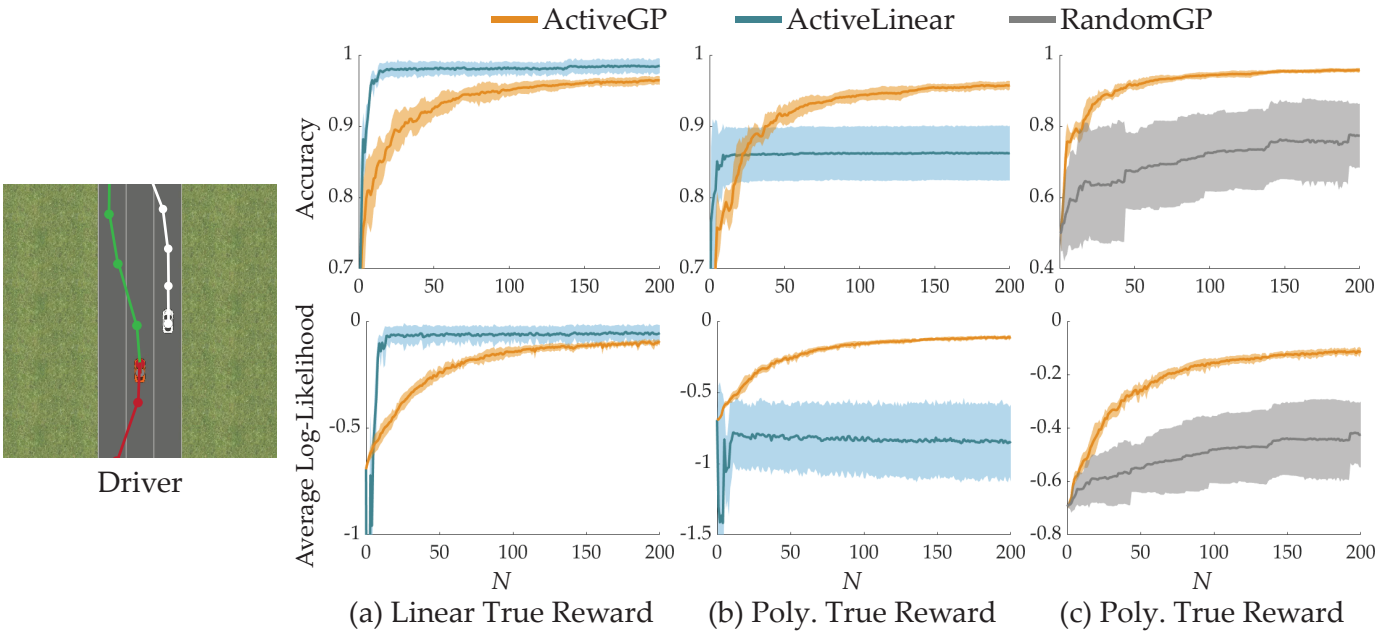


Fig. 3. Accuracies and average log-likelihoods for test set queries are shown for the *Driver* environment (mean±std over 5 runs). **(a)** Expressiveness results when the true underlying reward function is linear. **(b)** Expressiveness results when the true underlying reward function is a degree-of-two polynomial. **(c)** Data-efficiency results that compare ACTIVEGP with RANDOMGP. Accuracies and average log-likelihoods for test set queries are shown (mean±std). Active query generation improves data-efficiency over random querying in both tasks. This can be seen through both accuracy and log-likelihood.

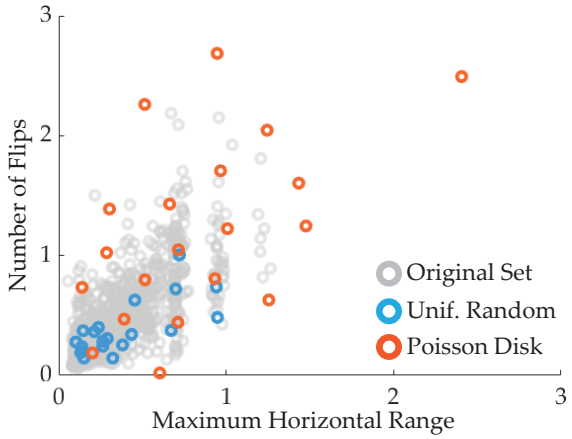


Fig. 4. Features of 1000 *Tosser* trajectories are visualized in two-dimensional plane (gray). Poisson disk sampling allows us to obtain a diverse set of 20 samples (orange), whereas sampling uniformly at random yields mostly uninteresting trajectories (blue).

- ACTIVEGP: The reward is modeled as a Gaussian Process. We will use our active query generation method to generate the most informative comparison queries to efficiently learn the reward function.

We generated a training dataset of trajectories with uniformly randomly selected actions. At every iteration of ACTIVEGP and ACTIVELINEAR, we computed the expected information gain of each possible query from this dataset to select the most informative query. This approach decreases the computation time compared to solving a continuous optimization over all possible trajectories as it was done by [12], [20].

C. Evaluation

We compare GP reward with linear reward in terms of *expressiveness* (ACTIVEGP vs. ACTIVELINEAR), and compare active query generation with random querying baseline in terms of *data-efficiency* (ACTIVEGP vs. RANDOMGP).

Test Set Generation. For both analyses on the expressiveness and data-efficiency, we also generated test sets of trajectories from the same distribution as the training set. However, it would not be fair to use the test set as is. Obtained with uniformly random action sequences, the majority of the training set is uninteresting trajectories, e.g. the ego car moves slightly forward and backward (similar to a random walk) in *Driver*, or the robot does not hit the capsule in *Tosser*. Using the test set without further modifications would mean we give more importance to these uninteresting behaviors as they form the majority in the datasets. Obviously, this is not the case. We want to learn the reward function everywhere in the dynamically feasible region with equal importance.

Hence, we adopted Poisson disk sampling [48] to get a diverse set of trajectories from the test set. Poisson disk sampling makes sure the difference between trajectories[3] is above some threshold by rejecting the samples that violate this constraint. A small example set of samples is compared to uniformly random samples in Fig. 4 for the *Tosser* environment.

After obtaining the diverse test set, we stored the true (noiseless) response of the simulated user for each possible query in this set. For the analysis on expressiveness, we computed the accuracy and the log-likelihood of the true responses under the reward functions that are learned with N actively chosen queries (up to $N = 200$). For data-efficiency analysis, we again used the true human responses to the queries in the diverse test set (only from the polynomial reward functions) to calculate the accuracy and the log-likelihood under the learned reward functions.

[3]We used L_2 distance between the feature vectors.

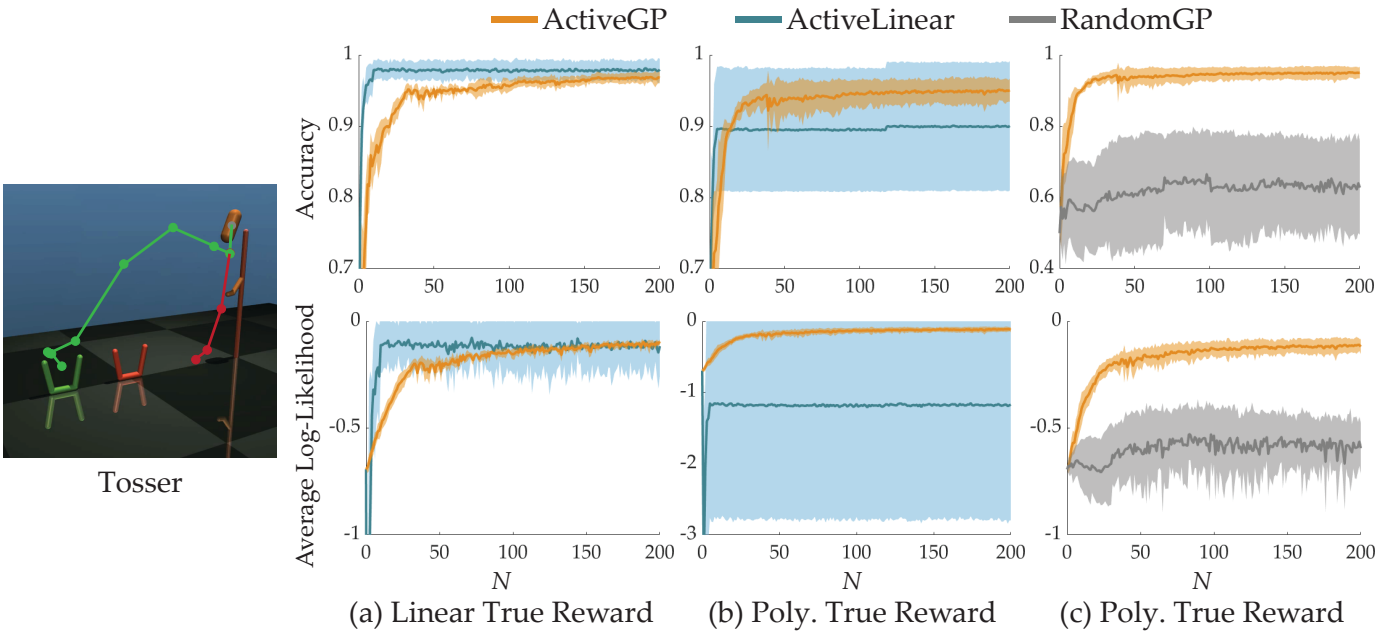


Fig. 5. Accuracies and average log-likelihoods for test set queries are shown for the *Tosser* environment (mean±std over 5 runs). **(a)** Expressiveness results when the true underlying reward function is linear. **(b)** Expressiveness results when the true underlying reward function is a degree-of-two polynomial. **(c)** Data-efficiency results that compare ACTIVEGP with RANDOMGP. Accuracies and average log-likelihoods for test set queries are shown (mean±std). Active query generation improves data-efficiency over random querying in both tasks. This can be seen through both accuracy and log-likelihood.

Expressiveness. Figs. 3(a,b) and 5(a,b) show the results of expressiveness simulations. When the true reward is polynomial, the linear model results in very high variance in both accuracy and likelihood, because its performance relies on how good a linear function can explain the true nonlinear reward. In this case, the GP model captures nonlinearities better than the linear model and provides better learning (Figs. 3(b) and 5(b)). When the true reward function is linear in features, a linear model naturally learns faster. However, as shown in Figs. 3(a) and 5(a), even in that case, GP model can achieve linear model's performance. To further improve the reward model, one can consider an approach to combine the linear and GP models by keeping a belief distribution over whether the true reward is linear or not, and actively querying the user according to this belief. We leave this extension as future work.

Data-Efficiency. We then evaluated how our active query generation helps with data-efficiency. Figs. 3(c) and 5(c) compare ACTIVEGP and RANDOMGP for the simulation environments. It can be seen that active querying significantly accelerates learning over random querying. It should be noted that the number of samples taken via Poisson disk sampling matters: While choosing a very small number will increase the variance in the results, choosing a very large number will make random querying seem like it performs comparable to (or even better than) the active querying as the test set will mostly consist of uninteresting trajectories, which are also abundant in the training set, as we stated earlier.

VI. USER STUDIES

A. Experiment Setup

We also compare our method ACTIVEGP with ACTIVELINEAR and RANDOMGP on a user study with a Fetch mobile manipulator robot [49]. In this study, the human subjects teach the Fetch robot how to play a variant of mini golf where the robot can achieve different scores by hitting the ball to different targets (see Fig. 1 and Fig. 6 for the set up). However, these scores are only known to the human. In fact, the robot does not even know the locations of the targets, and it tries to learn the reward as a function of its control inputs. Fixing some of the joints, we let the robot vary only its shot speed and angle, which are also the features of the reward function.

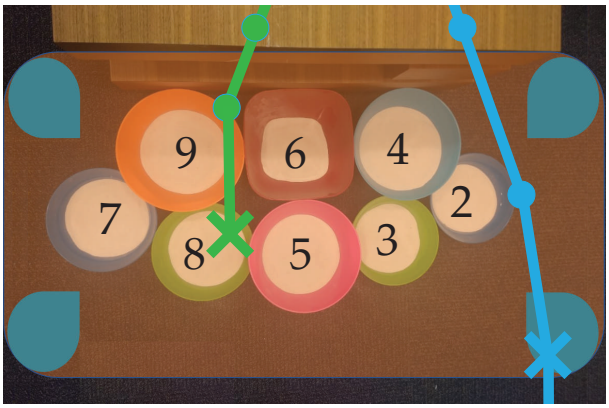


Fig. 6. Top view of the eight targets in the variant of mini-golf user study. The users assign distinct scores from 2 to 9 to the targets. The figure shows an example of this ranking. While the robot is capable of hitting the ball into the entire shaded region, the maximizers of a linear reward always lie near the corners of the shaded region in blue. Therefore, while the GP reward model can query the user with better trajectories (e.g. the green trajectory), the linear model only explores the boundaries (e.g. the blue trajectory that throws the ball outside of this region). Crosses show where the ball hits the ground.

This experiment setting is interesting because a linear reward function can only encode whether the robot must hit the ball to the right or to the left, or whether it must hit with high or low speed. It cannot particularly encourage (or discourage)

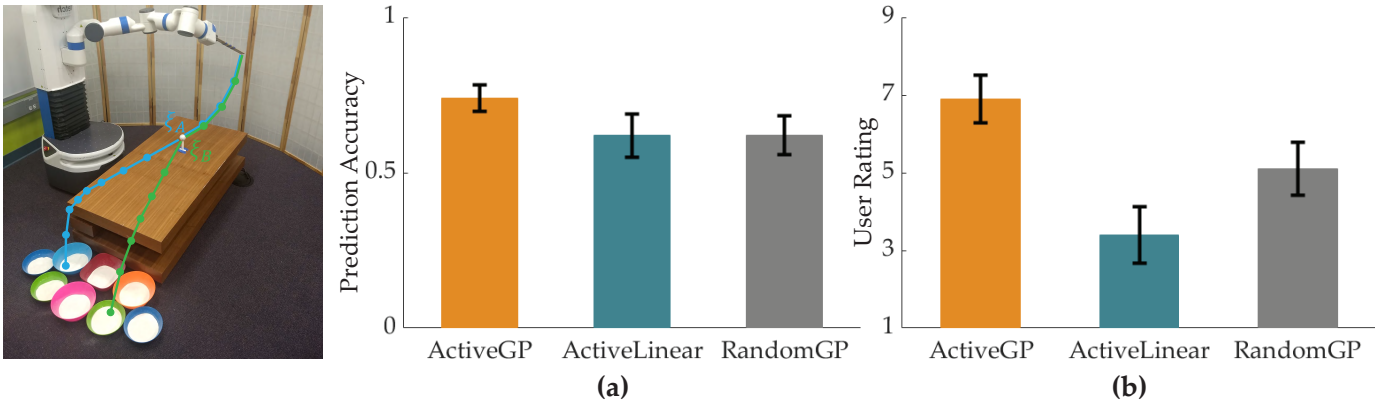


Fig. 7. **(a)** Prediction accuracy results (mean±se). Each trained with 15 queries, ACTIVEGP achieves significantly higher prediction accuracy than both ACTIVELINEAR and RANDOMGP ($p < 0.05$). **(b)** User ratings on the final robot performance (mean±se). ACTIVEGP accomplishes the task significantly better than both ACTIVELINEAR and RANDOMGP ($p < 0.05$).

hitting with a modest angle and/or speed. Therefore, as we show in Fig. 6, the targets that are around the middle region cannot be the maximizers of a linear reward function.

B. Subjects and Procedure

We recruited 10 users (6 males, 4 females) with an age range from 19 to 28. Each user first assigned their distinct scores (from 2 to 9) to the eight targets. The robot then queried them with 50 pairwise comparison questions: 15 for ACTIVEGP, 15 for ACTIVELINEAR, 15 for RANDOMGP and 5 queries generated uniformly at random to create a test set. We shuffled the order of queries to avoid any bias. We used the reward models, each of which is learned with 15 queries, to predict the user responses in the test set. The prediction score on the test set provides an accuracy metric.

In addition to the accuracy, we assessed whether the robot could successfully learn how to perform a good shot. For this, after the subjects responded to 50 queries, the robot demonstrated 3 more trajectories each of which correspond to the optimal trajectory of one method, the trajectory that maximizes the learned reward function. Again, the order of these trajectories was shuffled. After watching each demonstration, the subjects assigned a score to the shot from a 9-point Likert scale (1-very bad, 9-very good).

C. Results and Discussion

We provide a video that gives an overview of user studies and their results at https://youtu.be/SLSO2lBj9Mw.

Fig. 7(a) shows the prediction accuracy values on the test sets collected from the subjects (averaged over the subjects). By modeling the reward using a GP and querying the users with the most informative questions, ACTIVEGP achieves significantly higher prediction accuracy (0.74 ± 0.04, mean±se) compared to both ACTIVELINEAR (0.62 ± 0.07) and RANDOMGP (0.62 ± 0.06) with $p < 0.05$ (Wilcoxon signed rank test). The results from this user study are aligned with our simulation user studies.

In reward learning, it is crucial to validate whether the learned reward function can encode the desired behavior or not. Fig. 7(b) shows the user ratings to the trajectories that the robot showed after learning the user preferences via 3 different methods. ACTIVEGP obtains significantly higher scores (6.9 ± 0.6) than both ACTIVELINEAR (3.4 ± 0.7) and RANDOMGP (5.1 ± 0.7) with $p < 0.05$. While ACTIVELINEAR occasionally achieves high scores when the users' preferred target is near the edge, it generally fails to produce the desired behavior due to its low expressive power.

VII. CONCLUSION

Summary. We developed an active preference-based GP regression technique for reward learning. Our work tackles the lack of expressiveness of reward functions, data-inefficiency, and the incapability to demonstrate or quantify trajectories. Our results in simulations and user studies suggest our method is more successful in expressiveness and data-efficiency compared to the baselines.

Limitations and Future Work. We developed our methods only for pairwise comparisons. While extending them to learning from rankings is not mathematically very complicated, its data-efficiency compared to pairwise comparisons needs thorough analysis. Similarly, one could easily incorporate options to denote user uncertainty, which was shown to ease the process for humans [18]. GP regression becomes computationally heavy when the domain is high-dimensional (when d is large). This is a limitation of our work due to the use of GPs, and can be alleviated through efficient rank-one GP update approximations. Finally, although our methods ease the feature design, there still needs to be a design stage—it is often unrealistic to hope $\Psi(\xi) = \xi$ will work, due to high dimensionality of Ξ. Further research is warranted to simultaneously learn both the reward function f and the feature function Ψ.

ACKNOWLEDGMENTS

We thank Farid Soroush for the early discussions on active preference-based GP regression; Sydney M. Katz, Amir Maleki and Juan Carlos Aragon for the discussions on alternative ways to ease feature design. We acknowledge funding by Allstate, FLI grant RFP2-000, and NSF grant #1941722 and #1849952.

References

[1] J. Ho and S. Ermon, "Generative adversarial imitation learning," in *Advances in Neural Information Processing Systems (NIPS)*, 2016, pp. 4565–4573.

[2] S. Ross, N. Melik-Barkhudarov, K. S. Shankar, A. Wendel, D. Dey, J. A. Bagnell, and M. Hebert, "Learning monocular reactive uav control in cluttered natural environments," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2013, pp. 1765–1772.

[3] J. Song, H. Ren, D. Sadigh, and S. Ermon, "Multi-agent generative adversarial imitation learning," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2018, pp. 7472–7483.

[4] B. C. Stadie, P. Abbeel, and I. Sutskever, "Third-person imitation learning," in *International Conference on Learning Representations (ICLR)*, 2017.

[5] J. Clark and D. Amodei, *Faulty reward functions in the wild*, 2016. [Online]. Available: https://openai.com/blog/faulty-reward-functions/.

[6] A. Y. Ng, D. Harada, and S. Russell, "Policy invariance under reward transformations: Theory and application to reward shaping," in *International Conference on Machine Learning (ICML)*, vol. 99, 1999, pp. 278–287.

[7] P. F. Christiano, J. Leike, T. Brown, M. Martic, S. Legg, and D. Amodei, "Deep reinforcement learning from human preferences," in *Advances in Neural Information Processing Systems (NIPS)*, 2017, pp. 4299–4307.

[8] M. Cakmak, S. S. Srinivasa, M. K. Lee, J. Forlizzi, and S. Kiesler, "Human preferences for robot-human hand-over configurations," in *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2011, pp. 1986–1993.

[9] B. Ibarz, J. Leike, T. Pohlen, G. Irving, S. Legg, and D. Amodei, "Reward learning from human preferences and demonstrations in atari," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2018, pp. 8011–8023.

[10] D. S. Brown and S. Niekum, "Deep bayesian reward learning from preferences," in *NeurIPS Workshop on Safety and Robustness in Decision Making*, 2019.

[11] M. Tucker, E. Novoseller, C. Kann, Y. Sui, Y. Yue, J. Burdick, and A. D. Ames, "Preference-based learning for exoskeleton gait optimization," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2020.

[12] D. Sadigh, A. D. Dragan, S. S. Sastry, and S. A. Seshia, "Active preference-based learning of reward functions," in *Robotics: Science and Systems*, 2017.

[13] N. Wilde, D. Kulić, and S. L. Smith, "Bayesian active learning for collaborative task specification using equivalence regions," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 1691–1698, 2019.

[14] R. Akrour, M. Schoenauer, and M. Sebag, "April: Active preference learning-based reinforcement learning," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, 2012, pp. 116–131.

[15] J. R. Lepird, M. P. Owen, and M. J. Kochenderfer, "Bayesian preference elicitation for multiobjective engineering design optimization," *Journal of Aerospace Information Systems*, vol. 12, no. 10, pp. 634–645, 2015.

[16] E. Biyik and D. Sadigh, "Batch active preference-based learning of reward functions," in *Conference on Robot Learning (CoRL)*, 2018.

[17] E. Biyik, K. Wang, N. Anari, and D. Sadigh, "Batch active learning using determinantal point processes," *arXiv preprint arXiv:1906.07975*, 2019.

[18] E. Biyik, M. Palan, N. C. Landolfi, D. P. Losey, and D. Sadigh, "Asking easy questions: A user-friendly approach to active reward learning," in *Conference on Robot Learning (CoRL)*, 2019.

[19] C. Basu, E. Biyik, Z. He, M. Singhal, and D. Sadigh, "Active learning of reward dynamics from hierarchical queries," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019.

[20] M. Palan, N. C. Landolfi, G. Shevchuk, and D. Sadigh, "Learning reward functions by integrating human demonstrations and preferences," in *Robotics: Science and Systems*, 2019.

[21] S. M. Katz, A.-C. L. Bihan, and M. J. Kochenderfer, "Learning an urban air mobility encounter model from expert preferences," in *Digital Avionics Systems Conference (DASC)*, 2019.

[22] E. Biyik, D. A. Lazar, D. Sadigh, and R. Pedarsani, "The green choice: Learning and influencing human decisions on shared roads," in *IEEE Conference on Decision and Control (CDC)*, 2019.

[23] M. Racca, A. Oulasvirta, and V. Kyrki, "Teacher-aware active robot learning," in *ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, 2019, pp. 335–343.

[24] P. Abbeel and A. Y. Ng, "Apprenticeship learning via inverse reinforcement learning," in *International Conference on Machine Learning (ICML)*, 2004, p. 1.

[25] ——, "Exploration and apprenticeship learning in reinforcement learning," in *International Conference on Machine Learning (ICML)*, 2005, pp. 1–8.

[26] A. Y. Ng and S. J. Russell, "Algorithms for inverse reinforcement learning," in *International Conference on Machine Learning (ICML)*, vol. 1, 2000, p. 2.

[27] D. Ramachandran and E. Amir, "Bayesian inverse reinforcement learning.," in *International Joint Conference on Artificial Intelligence (IJCAI)*, vol. 7, 2007, pp. 2586–2591.

[28] B. D. Ziebart, A. L. Maas, J. A. Bagnell, and A. K. Dey, "Maximum entropy inverse reinforcement learning.," in *AAAI Conference on Artificial Intelligence (AAAI)*, vol. 8, 2008, pp. 1433–1438.

[29] D. P. Losey, K. Srinivasan, A. Mandlekar, A. Garg, and D. Sadigh, "Controlling assistive robots with learned

latent actions," in *International Conference on Robotics and Automation (ICRA)*, 2020.

[30] B. Akgun, M. Cakmak, K. Jiang, and A. L. Thomaz, "Keyframe-based learning from demonstration," *International Journal of Social Robotics*, vol. 4, no. 4, pp. 343–355, 2012.

[31] A. D. Dragan and S. S. Srinivasa, "Formalizing assistive teleoperation," in *Robotics: Science and Systems*, 2012.

[32] S. Javdani, S. S. Srinivasa, and J. A. Bagnell, "Shared autonomy via hindsight optimization," in *Robotics: Science and Systems*, 2015.

[33] R. P. Khurshid and K. J. Kuchenbecker, "Data-driven motion mappings improve transparency in teleoperation," *Presence: Teleoperators and Virtual Environments*, vol. 24, no. 2, pp. 132–154, 2015.

[34] C. Basu, Q. Yang, D. Hungerman, M. Sinahal, and A. D. Dragan, "Do you want your autonomous car to drive like you?" In *ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, 2017, pp. 417–425.

[35] A. Bajcsy, D. P. Losey, M. K. O'Malley, and A. D. Dragan, "Learning robot objectives from physical human interaction," *Proceedings of Machine Learning Research*, vol. 78, pp. 217–226, 2017.

[36] ——, "Learning from physical human corrections, one feature at a time," in *ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, 2018, pp. 141–149.

[37] J. González, Z. Dai, A. Damianou, and N. D. Lawrence, "Preferential bayesian optimization," in *International Conference on Machine Learning (ICML)*, 2017, pp. 1282–1291.

[38] W. Chu and Z. Ghahramani, "Preference learning with gaussian processes," in *International Conference on Machine Learning (ICML)*, 2005, pp. 137–144.

[39] N. Houlsby, F. Huszar, Z. Ghahramani, and J. M. Hernández-Lobato, "Collaborative gaussian processes for preference learning," in *Advances in Neural Information Processing Systems (NIPS)*, 2012, pp. 2096–2104.

[40] B. S. Jensen and J. B. Nielsen, "Pairwise judgements and absolute ratings with gaussian process priors," *Technical University of Denmark (DTU), Department of Applied Mathematics and Computer Science, Tech. Rep., Nov*, 2011.

[41] A. Kapoor, K. Grauman, R. Urtasun, and T. Darrell, "Active learning with gaussian processes for object categorization," in *International Conference on Computer Vision (ICCV)*, 2007, pp. 1–8.

[42] N. Houlsby, F. Huszár, Z. Ghahramani, and M. Lengyel, "Bayesian active learning for classification and preference learning," in *NIPS Workshop on Bayesian optimization, experimental design and bandits: Theory and applications*, 2011.

[43] C. Daniel, O. Kroemer, M. Viering, J. Metz, and J. Peters, "Active reward learning with a novel acquisition function," *Autonomous Robots*, vol. 39, no. 3, pp. 389–405, 2015.

[44] C. E. Rasmussen and C. K. Williams, *Gaussian processes for machine learning*. MIT Press, 2005.

[45] H. Nickisch and C. E. Rasmussen, "Approximations for binary gaussian process classification," *Journal of Machine Learning Research*, vol. 9, no. Oct, pp. 2035–2078, 2008.

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

[47] E. Todorov, T. Erez, and Y. Tassa, "Mujoco: A physics engine for model-based control," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2012, pp. 5026–5033.

[48] R. Bridson, "Fast poisson disk sampling in arbitrary dimensions.," *SIGGRAPH Sketches*, vol. 10, pp. 1 278 780–1 278 807, 2007.

[49] M. Wise, M. Ferguson, D. King, E. Diehr, and D. Dymesich, "Fetch and freight: Standard platforms for service robot applications," in *Workshop on Autonomous Mobile Service Robots*, 2016.

VIII. Appendix

A. Active Query Generation Derivation

Let Σ be the posterior covariance matrix between $f(\Psi^{(1)})$ and $f(\Psi^{(2)})$. And let

$$\Sigma^{-1} = \begin{bmatrix} c & d \\ d & c' \end{bmatrix}.$$

Throughout the derivation, all integrals are calculated over $\mathbb{R}$, but we drop it to simplify the notation. We write the first entropy term in the optimization (7) as:

$$\begin{aligned}
H(q \mid \Psi, \boldsymbol{\Psi}, \boldsymbol{q}) &= h\left(\int\int \Phi\left(\frac{f^{(1)}-f^{(2)}}{\sqrt{2}\sigma}\right)\mathcal{N}([f^{(1)},f^{(2)}] \mid [\mu^{(1)},\mu^{(2)}],\Sigma)df^{(2)}df^{(1)}\right) \\
&= h\left(\frac{\sqrt{cc'-d^2}}{2\pi}\int\int \Phi\left(\frac{f^{(1)}-f^{(2)}}{\sqrt{2}\sigma}\right)e^{-\frac{1}{2}(c(f^{(1)}-\mu^{(1)})^2+c'(f^{(2)}-\mu^{(2)})^2+2d(f^{(1)}-\mu^{(1)})(f^{(2)}-\mu^{(2)}))}df^{(2)}df^{(1)}\right) \\
&= h\left(\frac{\sqrt{cc'-d^2}}{2\pi}\int\int \Phi\left(\frac{f^{(1)}-f^{(2)}}{\sqrt{2}\sigma}\right)e^{-\frac{1}{2}(c((f^{(1)}-\mu^{(1)})^2+\frac{2d}{c}(f^{(1)}-\mu^{(1)})(f^{(2)}-\mu^{(2)}))+c'(f^{(2)}-\mu^{(2)})^2)}df^{(1)}df^{(2)}\right) \\
&= h\left(\frac{\sqrt{cc'-d^2}}{2\pi}\int\int \Phi\left(\frac{f^{(1)}-f^{(2)}}{\sqrt{2}\sigma}\right)e^{-\frac{1}{2}(c(f^{(1)}-\mu^{(1)}+\frac{d}{c}(f^{(2)}-\mu^{(2)}))^2-\frac{d^2}{c}(f^{(2)}-\mu^{(2)})^2+c'(f^{(2)}-\mu^{(2)})^2)}df^{(1)}df^{(2)}\right) \\
&= h\left(\frac{\sqrt{cc'-d^2}}{2\pi}\int e^{-\frac{1}{2}c'(f^{(2)}-\mu^{(2)})^2}e^{\frac{1}{2}\frac{d^2}{c}(f^{(2)}-\mu^{(2)})^2}\int \Phi\left(\frac{f^{(1)}-f^{(2)}}{\sqrt{2}\sigma}\right)e^{-\frac{1}{2}(c(f^{(1)}-\mu^{(1)}+\frac{d}{c}(f^{(2)}-\mu^{(2)}))^2)}df^{(1)}df^{(2)}\right) \\
&= h\left(\frac{\sqrt{cc'-d^2}}{2\pi}\int e^{-\frac{1}{2}\frac{c'c-d^2}{c}(f^{(2)}-\mu^{(2)})^2}\int \frac{\Phi\left(\frac{f^{(1)}-f^{(2)}}{\sqrt{2}\sigma}\right)e^{-\frac{1}{2}(c(f^{(1)}-\mu^{(1)}+\frac{d}{c}(f^{(2)}-\mu^{(2)}))^2)}}{\frac{\sqrt{2\pi}}{\sqrt{c}}}\frac{\sqrt{2\pi}}{\sqrt{c}}df^{(1)}df^{(2)}\right) \\
&= h\left(\frac{\sqrt{cc'-d^2}}{\sqrt{2\pi}\sqrt{c}}\int e^{-\frac{1}{2}\frac{c'c-d^2}{c}(f^{(2)}-\mu^{(2)})^2}\int \frac{\Phi\left(\frac{f^{(1)}-f^{(2)}}{\sqrt{2}\sigma}\right)e^{-\frac{1}{2}(c(f^{(1)}-\mu^{(1)}+\frac{d}{c}(f^{(2)}-\mu^{(2)}))^2)}}{\frac{\sqrt{2\pi}}{\sqrt{c}}}df^{(1)}df^{(2)}\right)
\end{aligned}$$

Using the mathematical identity $\int_x \phi(x)N(x|\mu,\sigma^2)dx = \phi(\frac{\mu}{\sqrt{1+\sigma^2}})$, we obtain

$$\begin{aligned}
H(q \mid \Psi, \boldsymbol{\Psi}, \boldsymbol{q}) &= h\left(\frac{\sqrt{cc'-d^2}}{\sqrt{2\pi}\sqrt{c}}\int e^{-\frac{1}{2}\frac{c'c-d^2}{c}(f^{(2)}-\mu^{(2)})^2}\Phi\left(\frac{\mu^{(1)}-\frac{d}{c}f^{(2)}+\frac{d}{c}\mu^{(2)}-f^{(2)}}{\sqrt{2}\sigma\sqrt{1+\frac{1}{2c\sigma^2}}}\right)df^{(2)}\right) \\
&= h\left(\frac{\sqrt{cc'-d^2}}{\sqrt{2\pi}\sqrt{c}}\int e^{-\frac{1}{2}\frac{c'c-d^2}{c}(f^{(2)}-\mu^{(2)})^2}\Phi\left(\frac{\mu^{(1)}+\frac{d}{c}\mu^{(2)}-(\frac{d}{c}+1)f^{(2)}}{\sqrt{2}\sigma\sqrt{1+\frac{1}{2c\sigma^2}}}\right)df^{(2)}\right) \\
&= h\left(\frac{\sqrt{cc'-d^2}}{\sqrt{2\pi}\sqrt{c}}\int e^{-\frac{1}{2}\frac{c'c-d^2}{c}(f^{(2)}-\mu^{(2)})^2}\frac{\Phi\left(\frac{-(\frac{d}{c}+1)(f^{(2)}-\frac{\mu^{(1)}+\frac{d}{c}\mu^{(2)}}{\frac{d}{c}+1})}{\sqrt{2}\sigma\sqrt{1+\frac{1}{2c\sigma^2}}}\right)}{\frac{\sqrt{2\pi c}}{\sqrt{c'c-d^2}}}\frac{\sqrt{2\pi c}}{\sqrt{c'c-d^2}}df^{(2)}\right)
\end{aligned}$$

Using the same identity again,

$$H(q \mid \Psi, \boldsymbol{\Psi}, \boldsymbol{q}) = h\left(\Phi\left(\frac{\mu^{(1)}-\mu^{(2)}}{\sqrt{2}\sigma\sqrt{1+\frac{1}{2c\sigma^2}}\sqrt{1+\frac{c}{2\sigma^2+\frac{1}{c}}\frac{(1+\frac{d}{c})^2}{c'c-d^2}}}\right)\right)$$

One can then expand the expression in the denominator and use the facts that $\mathrm{Var}(f(\Psi^{(1)})) = \frac{c'}{cc'-d^2}$, $\mathrm{Var}(f(\Psi^{(2)})) = \frac{c}{cc'-d^2}$ and $\mathrm{Cov}(f(\Psi^{(1)}), f(\Psi^{(2)})) = \frac{-d}{cc'-d^2}$ to obtain

$$H(q \mid \Psi, \boldsymbol{\Psi}, \boldsymbol{q}) = h\left(\Phi\left(\frac{\mu^{(1)}-\mu^{(2)}}{\sqrt{2\sigma^2+g(\Psi^{(1)},\Psi^{(2)})}}\right)\right).$$

where $g(\Psi^{(1)},\Psi^{(2)}) = \mathrm{Var}(f(\Psi^{(1)})) + \mathrm{Var}(f(\Psi^{(2)})) - 2\mathrm{Cov}(f(\Psi^{(1)}), f(\Psi^{(2)}))$

We next make the derivation for the second entropy term. To simplify the notation, we let $\sigma'^2 = \frac{\pi ln(2)}{2}, \sigma''^2 = \sigma'^2 + \frac{1}{c}$, and $\sigma_b^2 = \frac{c(1+\frac{d}{c})^2}{c'c-d^2}$. By performing a linearization over the logarithm of the second entropy term as in [42],

$$\begin{aligned}
\mathbb{E}_{f\sim P(f|\mathbf{\Psi},\boldsymbol{q})}\left[H(q \mid \Psi, f)\right] &\approx \frac{\sqrt{c'c-d^2}}{2\pi}\int\int e^{-\frac{(f^{(1)}-f^{(2)})^2}{\pi ln(2)}} e^{-\frac{1}{2}(c(f^{(1)}-\mu^{(1)})^2+c'(f^{(2)}-\mu^{(2)})^2+2d(f^{(1)}-\mu^{(1)})(f^{(2)}-\mu^{(2)}))} df^{(1)} df^{(2)} \\
&= \frac{\sqrt{c'c-d^2}}{2\pi}\int e^{-\frac{1}{2}c'(f^{(2)}-\mu^{(2)})^2}\int e^{-\frac{(f^{(1)}-f^{(2)})^2}{2\sigma'^2}} e^{-\frac{1}{2}(c(f^{(1)}-\mu^{(1)})^2+2d(f^{(1)}-\mu^{(1)})(f^{(2)}-\mu^{(2)}))} df^{(1)} df^{(2)} \\
&= \frac{\sqrt{c'c-d^2}}{2\pi}\int e^{-\frac{1}{2}c'f^{(2)^2}}\int e^{-\frac{(f^{(1)}+\mu^{(1)}-f^{(2)}-\mu^{(2)})^2}{2\sigma'^2}} e^{-\frac{1}{2}cf^{(1)^2}-df^{(1)}f^{(2)}} df^{(1)} df^{(2)} \\
&= \frac{\sqrt{c'c-d^2}}{2\pi}\int e^{-\frac{1}{2}c'f^{(2)^2}}\int e^{-\frac{(f^{(1)}+\mu^{(1)}-f^{(2)}-\mu^{(2)})^2}{2\sigma'^2}} e^{-\frac{1}{2}c(f^{(1)}+\frac{d}{c}f^{(2)})^2+\frac{1}{2}\frac{d^2}{c}f^{(2)^2}} df^{(1)} df^{(2)} \\
&= \frac{\sqrt{c'c-d^2}}{2\pi}\int e^{-\frac{1}{2}\frac{c'c-d^2}{c}f^{(2)^2}}\int e^{-\frac{(f^{(1)}-f^{(2)}+\mu^{(1)}-\mu^{(2)})^2}{2\sigma'^2}} e^{-\frac{1}{2}c(f^{(1)}+\frac{d}{c}f^{(2)})^2} df^{(1)} df^{(2)}
\end{aligned}$$

By the change of variables for the inner integral with $u = f^{(1)} + \frac{d}{c}f^{(2)}$,

$$\begin{aligned}
\mathbb{E}_{f\sim P(f|\mathbf{\Psi},\boldsymbol{q})}\left[H(q \mid \Psi, f)\right] &= \frac{\sqrt{c'c-d^2}}{2\pi}\int e^{-\frac{1}{2}\frac{c'c-d^2}{c}f^{(2)^2}}\int_u e^{-\frac{(u-\frac{d}{c}f^{(2)}+\mu^{(1)}-f^{(2)}-\mu^{(2)})^2}{2\sigma'^2}} e^{-\frac{1}{2}cu^2} du df^{(2)} \\
&= \frac{\sqrt{c'c-d^2}}{2\pi}\int e^{-\frac{1}{2}\frac{c'c-d^2}{c}f^{(2)^2}}\int_u e^{-\frac{((1+\frac{d}{c})f^{(2)}-u-\mu^{(1)}+\mu^{(2)})^2}{2\sigma'^2}} e^{-\frac{1}{2}cu^2} du df^{(2)}
\end{aligned}$$

By another change of variables for the outer integral with $v = \frac{f^{(2)}}{1+d/c}$,

$$\mathbb{E}_{f\sim P(f|\mathbf{\Psi},\boldsymbol{q})}\left[H(q \mid \Psi, f)\right] = \frac{1}{1+\frac{d}{c}}\frac{\sqrt{c'c-d^2}}{2\pi}\int_v e^{-\frac{1}{2}\frac{c'c-d^2}{c}\frac{v^2}{(1+\frac{d}{c})^2}}\int_u e^{-\frac{(v-u+\mu^{(2)}-\mu^{(1)})^2}{2\sigma'^2}} e^{-\frac{1}{2}cu^2} du dv$$

By identifying the inner integral as a convolution of two Gaussians, we get

$$\begin{aligned}
\mathbb{E}_{f\sim P(f|\mathbf{\Psi},\boldsymbol{q})}\left[H(q \mid \Psi, f)\right] &= \frac{1}{1+\frac{d}{c}}\frac{\sqrt{c'c-d^2}}{2\pi}2\pi\sigma'\frac{1}{\sqrt{c}}\int_v e^{-\frac{1}{2}\frac{c'c-d^2}{c}\frac{v^2}{(1+\frac{d}{c})^2}}\frac{1}{\sqrt{2\pi}\sqrt{\sigma'^2+\frac{1}{c}}}e^{-\frac{1}{2}\frac{(v-(\mu^{(1)}-\mu^{(2)}))^2}{\sigma'^2+\frac{1}{c}}} dv \\
&= \frac{1}{1+\frac{d}{c}}\sqrt{c'c-d^2}\sigma'\frac{1}{\sqrt{c}}\frac{1}{\sqrt{2\pi}\sqrt{\sigma'^2+\frac{1}{c}}}\int_v e^{-\frac{1}{2}\frac{v^2}{\sigma_b^2}}e^{-\frac{1}{2}\frac{(v-(\mu^{(1)}-\mu^{(2)}))^2}{\sigma''^2}} dv
\end{aligned}$$

By repeating the same convolution trick for the second integral,

$$\begin{aligned}
\mathbb{E}_{f\sim P(f|\mathbf{\Psi},\boldsymbol{q})}\left[H(q \mid \Psi, f)\right] &= \frac{1}{1+\frac{d}{c}}\sqrt{c'c-d^2}\sigma'\frac{1}{\sqrt{c}}\frac{1}{\sqrt{2\pi}\sqrt{\sigma'^2+\frac{1}{c}}}2\pi\sigma_b\sigma''\frac{1}{\sqrt{2\pi}\sqrt{\sigma_b^2+\sigma''^2}}e^{-\frac{1}{2}\frac{(\mu^{(1)}-\mu^{(2)})^2}{\sigma_b^2+\sigma''^2}} \\
&= \frac{1}{1+\frac{d}{c}}\sqrt{c'c-d^2}\sigma'\frac{1}{\sqrt{c}}\frac{1}{\sqrt{\sigma'^2+\frac{1}{c}}}\sigma_b\sigma''\frac{1}{\sqrt{\sigma_b^2+\sigma''^2}}e^{-\frac{1}{2}\frac{(\mu^{(1)}-\mu^{(2)})^2}{\sigma_b^2+\sigma''^2}}
\end{aligned}$$

Again, we express this in terms of covariance and variance expressions:

$$\mathbb{E}_{f\sim P(f|\mathbf{\Psi},\boldsymbol{q})}\left[H(q \mid \Psi, f)\right] = \frac{\sqrt{\pi\ln(2)\sigma^2}\exp\left(-\frac{(\mu_a-\mu b)^2}{\pi\ln(2)\sigma^2+2g(\Psi^{(1)},\Psi^{(2)})}\right)}{\sqrt{\pi\ln(2)\sigma^2+2g(\Psi^{(1)},\Psi^{(2)})}}$$