Sample and Feedback Efficient Hierarchical Reinforcement Learning from Human Preferences

Robert Pinsler[1] and Riad Akrour[2] and Takayuki Osa[3] and Jan Peters[2] and Gerhard Neumann[2,4]

***Abstract*— While reinforcement learning has led to promising results in robotics, defining an informative reward function can sometimes prove to be challenging. Prior work considered including the human in the loop to jointly learn the reward function and the optimal policy. Generating samples from a physical robot and requesting human feedback are both taxing efforts for which efficiency is critical. In contrast to prior work, in this paper we propose to learn reward functions from both the robot and the human perspectives in order to improve on both efficiency metrics. On one side, learning a reward function from the human perspective increases feedback efficiency by assuming that humans rank trajectories according to an outcome space of reduced dimensionaltiy. On the other side, learning a reward function from the robot perspective circumvents the need for learning a dynamics model while retaining the sample efficiency of model-based approaches. We provide an algorithm that incorporates bi-perspective reward learning into a general hierarchical reinforcement learning framework and demonstrate the merits of our approach on a toy task and a simulated robot grasping task.**

I. INTRODUCTION

Autonomous robots equipped with dexterous manipulation skills yield great potential for improving our lives. For example, household robots could empty the dish washer or hand over everyday items, agricultural robots could pick vegetables, and disaster robots could segregate waste. Such robots need not only to grasp known objects but generalize their knowledge to unseen ones. While robotic grasping has been a long-standing research area [1], [2], grasping arbitrary objects remains a challenging problem for which data-driven approaches constitute a promising direction.

Data-driven approaches address the issue of generalization by utilizing prior grasp experience to synthesize grasps for unseen objects. While most of these approaches assume that a complete grasp database is given, reinforcement learning (RL) allows to collect grasp experiences from trial and error. However, learning a grasping policy through RL requires to specify a reward function. Most of the proposed quality measures were motivated by grasp analysis in simulation. But the outcome of a grasp in a real dynamic and noisy environment can be very different from simulation. In fact, Balasubramanian et al. [3] showed that grasps produced by optimizing such traditional grasp quality measures performed worse than kinesthetic teach-in on real systems. Moreover, Roa and Suárez [4] point out that there is no single measure suitable for every situation because those metrics only encode a subset of criteria defining a good grasp, such as stability or dexterity. While more complex reward functions could be engineered, such reward shaping practices can lead to undesirable robot behavior [5]. These findings suggest that designing a reward function by hand is not trivial.

Alternatively, inverse reinforcement learning [6], [7] can be used to recover the underlying reward function from expert demonstrations. Demonstrations are usually provided by humans through tele-operation or kinesthetic teaching. Presenting optimal or near-optimal demonstrations can however be challenging without prior training. For example, differences between the human hand and robot hardware complicate the generation of human demonstrations for the robot grasping task. To lighten the assumption on the expertise of the human teacher, we consider instead to learn the reward function from human preferences [8]. In learning from human preferences, the human assesses the quality of demonstrations executed by the robot (e.g. attempts at grasping an object) instead of providing the demonstrations themselves. Humans are particularly apt at comparing items [9], [10], making preference feedback more reliable than absolute feedback. We take advantage of this fact by modeling the reward function as a latent function to be inferred by preference learning.

The main contributions of this paper are two-fold. First, we employ a novel reward learning scheme from human preferences that combines the perspectives of the human and the robot. Taking both perspectives into consideration allows to improve on both sample and feedback efficiency at the same time. From the robot's perspective, sample efficiency can be greatly improved if the expected reward of a decision (e.g. a particular motion parameter) can be predicted before it is executed. On the other side, the human teacher assesses the *outcome* of executing the motion parameters, not the motion parameters themselves. Such an outcome (e.g. how much the grasped object moved or how 'natural' the grasping appeared) is of significantly lower dimension than the motion parameters. Thus, we will learn from human preferences on the lower-dimensional outcome space and back-propagate information on the human's preferences to a reward model defined on the motion parameter space. Feedback efficiency can be further increased by only querying the human if the preference is expected to improve the model.

The second contribution of this paper is to incorporate the proposed bi-perspective reward learning scheme into a

[1] Technische Universität Darmstadt, 64289 Darmstadt, Germany
robert.pinsler@stud.tu-darmstadt.de
[2] Institut für Intelligente Autonome Systeme, Technische Universität Darmstadt, 64289 Darmstadt, Germany
{akrour, neumann}@ias.tu-darmstadt.de
[3] Department of Complexity Science and Engineering, University of Tokyo, Bunkyō-ku, Tōkyō, Japan
osa@edu.k.u-tokyo.ac.jp

Preprint submitted to 2018 IEEE International Conference on Robotics and Automation. Received September 15, 2017.

hierarchical reinforcement learning (HRL) framework. We chose the HRL framework for its generality and its ability to tackle complex learning problems. For instance, a task such as robot grasping requires to cover a broad range of motions. Prior knowledge of the task can be incorporated by imposing a hierarchy where first a grasp type is selected (e.g. pinch grasp or power grasp), which is then used to generate a grasping motion. Such a decomposition will greatly dampen the data requirements. In Sec. III, we will additionally show how the reward models from the robot's perspective can be used in different levels of the hierarchy to efficiently solve the exploitation-exploration dilemma.

II. RELATED WORK

Learning from human feedback has been extensively studied in the RL community. The tackled problems range from classical RL benchmarks with tabular or linear-in-features policies [11]–[16] to more complex environments requiring the use of deep neural networks [17]. While special care is taken in some of these approaches to reduce the amount of human queries [14]–[16], the number of rollouts generated by the learning agent is relatively high and would prohibit the application of any of these methods to physical robots.

To learn under a more restrictive sample constraint, the usual trade-off in robotics is to replace general purpose high-dimensional policies with specialized low-dimensional ones. In [18], global Bayesian optimization (BO) from human preferences is used to tune the control gains of a controller. However, such a global approach is known to not scale favorably and experiments were limited to three dimensional search spaces. In [19], a similar preference Gaussian Process (GP) [20] is used to model the human preferences but a local, more scalable policy search method is employed in order to optimize the policy parameters. However, as in [18], the human preference model is learned on the same parametric space of the policy. While it is appealing to do so as it allows the robot to predict the expected reward of a policy without generating additional rollouts, it has the drawback of requiring an excessively high number of feedback to learn the human preference model since GPs with the usual kernels are poor extrapolators (see [21], Sec. III.E.3).

Perhaps the closest work to ours is that of [22], which also learns the human preferences on the outcome space (which is significantly lower dimensional than the policy parameter space), employs local policy search to optimize the policy and tackles the problem of grasping for which a reward function is hard to hand-define. The key differences are in i) the use of preference feedback instead of absolute feedback (scores)—for which existing evidence suggest that it is in general a more robust human feedback [9], [10] and that in particular, humans prefer to give preference feedback when teaching robotic tasks [23], ii) the best of two worlds combination of [22] and [19] by back-propagating the reward model learned on the outcome space to a reward model on the parameter space in order to improve on both feedback and sample efficiency. The latter reward model is then used in iii) the hierarchical reinforcement learning scheme, allowing us to tackle more complex grasp problem than that of [22] where only a single grasp type was considered.

As for data-driven approaches to robot grasping, several authors have already considered the use of RL. Stulp et al. [24] employ RL to learn the goal and shape parameters of a movement primitive. They achieve robust grasps by taking position uncertainty into consideration. Krömer et al. [25] propose a hierarchical architecture comprising an upper-level reinforcement learner for predicting the grasp location and a low-level reactive controller that generates the grasp motion. Osa et al. [26] instead propose a hierarchical RL architecture that is able to generalize grasps to multiple grasp types and objects. We use the same basic architecture as Osa et al. [26], but propose a novel way to learn the rewards.

III. APPROACH

In a contextual policy search setting [27] our goal is to find the policy π, defining a distribution $\pi(\boldsymbol{\omega}|\mathbf{s})$ over motion parameters $\boldsymbol{\omega} \in \Omega$ (e.g. the goal position of a dynamical movement primitive [28]) for a given context $\mathbf{s} \in S$ (e.g. a description of the object to grasp), maximizing the expected return $J(\pi) = \int R_{\mathbf{s},\boldsymbol{\omega}} p(\mathbf{s})\pi(\boldsymbol{\omega}|\mathbf{s})$; where $R_{\mathbf{s},\boldsymbol{\omega}}$ is the reward for executing $\boldsymbol{\omega}$ in $\mathbf{s}$.

Learning in this paper relies on three working hypotheses within the general policy search framework: i) the reward function is not assumed to be known beforehand but learned from human preferences, while structural assumptions on ii) the action and iii) the context spaces are introduced in order to reduce the sample complexity of high-dimensional robotics applications. On the action space, we assume that a hierarchical decomposition of the policy is available such that the policy can be written as the mixture of K lower level policies of simpler shape, $\pi(\boldsymbol{\omega}|\mathbf{s}) = \sum_{k=1}^{K} \pi_u(k|\mathbf{s})\pi_l^k(\boldsymbol{\omega}|\mathbf{s})$; where π_u is the upper level policy or the gating and $\left\{\pi_l^k\right\}_{k=1}^{K}$ is a set of lower level policies (also referred to as sub-policies or options). This structure is widespread in the hierarchical reinforcement learning (HRL) literature [29]–[31].

On the context space, we assume the availability of a function $\mathcal{S}$ that takes as input a context $\tilde{\mathbf{s}}$ and returns a set of local contexts—typically of reduced dimensionality—$\{\mathbf{s}_i\}_{i=1}^{N}$ and over which the lower level policies are defined. Such assumption does not alter the generality of the algorithm as $\mathcal{S}$ can simply return $\{\tilde{\mathbf{s}}\}$. A more interesting function $\mathcal{S}$ is described in Sec. IV-B for the grasping problem. From here on, the global context of the task will be denoted by $\tilde{\mathbf{s}}$ while a local context over which sub-policies are defined is denote by s.

Algorithm 1 provides an outline of the approach. At the start of each episode, a global context $\tilde{\mathbf{s}} \sim \mu(\tilde{\mathbf{s}})$, specifying the task, is observed. The upper level policy, implicitly defined in line 4 to 8 of Alg. 1 then selects the most promising pair of local context s^* and sub-policy k^* within the set $\mathcal{S}(\tilde{\mathbf{s}}) \times K$. The upper level policy, further described in Sec. III-B.1, trades-off exploitation and exploration based on the expected reward of choosing option k in local context s and its associated variance. Note that both these estimates are computed from the context-parameter reward models

Preprint submitted to 2018 IEEE International Conference on Robotics and Automation. Received September 15, 2017.

Algorithm 1 Hierarchical Reinforcement Learning with Bi-Perspective Reward Learning from Preferences

1: **Input:** Option count K, active learning threshold λ
2: **repeat**
3: Observe global context $\tilde{\mathbf{s}} \sim \mu(\tilde{\mathbf{s}})$
4: **for all** $(\mathbf{s}, k) \in \mathcal{S}(\tilde{\mathbf{s}}) \times K$ **do**
5: $\mu_{s,k} = \mathbb{E}[R_{sw}|\mathcal{D}^k, \mathbf{s}, \pi_l^k]$
6: $\sigma^2_{s,k} = \mathrm{Var}[R_{sw}|\mathcal{D}^k, \mathbf{s}, \pi_l^k]$
7: **end for**
8: $\mathbf{s}^*, k^* = \arg\max_{\mathbf{s},k} \text{CGP-UCB}\,(\mu_{s,k}, \sigma_{s,k})$
9: Rollout $\boldsymbol{\tau} \sim p(\boldsymbol{\tau}|\mathbf{s}^*, \boldsymbol{\omega})$ with $\boldsymbol{\omega} \sim \pi_l^{k^*}(\boldsymbol{\omega}|\mathbf{s}^*)$
10: Observe outcome $\mathbf{o} = \phi(\boldsymbol{\tau})$
11: **if** $\mathbb{E}_{\succ}[\mathrm{KL}(p(\mathbf{R_o}|\mathcal{D}_{\succ+})||p(\mathbf{R_o}|\mathcal{D}_{\succ})] > \lambda$ **then**
12: Query human for preference feedback
13: **end if**
14: Update reward models
15: Update lower-level policies π_l^k
16: **until** Task learned
17: **return** Learned policies π_l^k and reward models

(reward from the robot perspective) and do not necessitate the generation of rollouts on the robot. The learning procedure for these reward models is described in Sec. III-A.1.

The selected sub-policy samples a motion parameter $\boldsymbol{\omega} \sim \pi_l^{k^*}(\boldsymbol{\omega}|\mathbf{s}^*)$, and a trajectory $\boldsymbol{\tau}$ is obtained by performing a rollout. This trajectory is further compressed by applying the feature function ϕ containing aspects of the trajectory the human is likely basing their feedback on. In line 11, we decide whether to ask for human feedback based on the expected change of the distribution of the outcome reward (reward model from the human perspective) had the feedback been requested. If a feedback is requested, the outcome reward model is update, whereas the context-parameter reward models are always updated upon the generation of a trajectory. Learning the outcome reward model and the active selection criterion is further described in Sec. III-A.2. Finally the lower level policies are updated as described in Sec. III-B.2.

A. Reward Learning

The selection of $(\mathbf{s}^*, k^*)$ depends on the expected reward $\mathbb{E}[R_{sw}|\mathcal{D}^k, \mathbf{s}, \pi_l^k]$ of each local context-option pair and their variance. A sample estimate of both quantities can be obtained by performing rollouts but would dramatically increase the sample complexity of the algorithm due to the high number of local context-option combinations. Instead, we will frame a regression problem and learn a probabilistic context-parameter reward model $p(R_{s\omega}|\mathcal{D}^k, \mathbf{s}, \boldsymbol{\omega})$. As the true reward is unknown, the targets of the regression problem are given by a second probabilistic model $p(R_o|\mathcal{D}_{\succ}, \mathbf{o})$, modeling the human preferences on the outcome space.

The outcome $o = \phi(\tau)$ is defined as a compression of a trajectory τ by the hand defined function ϕ. Intuitively, humans judge the quality of a trajectory on a small subset of effects the robot has caused on its environment instead of the internal parametric representations of the context and the planned motion of the robot. Learning on such a space drastically improves the feedback efficiency at the expense of manually defining the compression function ϕ.

1) Context-Parameter Reward Model: We learn a probabilistic reward model $p(R_{s\omega}|\mathcal{D}^k, \mathbf{s}, \boldsymbol{\omega})$ for each policy π_l^k from dataset $\mathcal{D}^k = \{\mathbf{s}_i, \boldsymbol{\omega}_i, \mathbf{o}_i, \bar{R}(\mathbf{o}_i)\}_{i=1:N_k}$, where $\bar{R}(\mathbf{o}) = \mathbb{E}[R_o|\mathcal{D}_{\succ}, \mathbf{o}]$ is the predicted mean reward for outcome $\mathbf{o}$ as provided by the outcome reward model. Letting $\mathbf{x} = (\mathbf{s}, \boldsymbol{\omega})$ denote the context-parameter pair, we note that even if $\bar{R}(\mathbf{o})$ is the true reward (instead of being estimated from human feedback), it would still be a noisy estimate of $R(\mathbf{x})$ due to the environment noise. As such we assume that the regression targets are corrupted by a Gaussian noise of deviation σ_n. The magnitude of σ_n depends on the stochasticity of the environment and will be estimated by hyper-parameter tuning. We use GP regression [32] to model the reward function, i.e.

$$R(\mathbf{x}) \sim \mathrm{GP}(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}')), \tag{1}$$

where the mean function $m(\cdot)$ is assumed to be zero and the covariance function $k(\mathbf{x}, \mathbf{x}')$ is given by the squared exponential covariance function,

$$k_{\mathrm{SE}}(\mathbf{x}, \mathbf{x}') = \sigma_f^2 \exp\left((\mathbf{x} - \mathbf{x}')^T \mathbf{M} (\mathbf{x} - \mathbf{x}')\right), \tag{2}$$

where $\mathbf{M} = \mathrm{diag}(\boldsymbol{\ell})$ is a diagonal matrix containing positive bandwidth parameters $\boldsymbol{\ell}$, which determine the contribution of each dimension of the input. The hyper-parameters $\boldsymbol{\theta} = \{\sigma_f, \sigma_n, \boldsymbol{\ell}\}$ are estimated by maximizing the marginal likelihood [32]. The latent rewards $\mathbf{r} = [R(\mathbf{x}_1), \ldots, R(\mathbf{x}_{N_k})]$ of the input data and the latent reward $R(\mathbf{x}_*)$ at a test point $\mathbf{x}_*$ are jointly Gaussian distributed,

$$\begin{bmatrix} \mathbf{r} \\ R(\mathbf{x}_*) \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} \mathbf{K} & \mathbf{k}_* \\ \mathbf{k}_*^T & k_{**} \end{bmatrix}\right), \tag{3}$$

where the ij-th element of the $N_k \times N_k$ matrix $\mathbf{K}$ is defined as $k(\mathbf{x}_i, \mathbf{x}_j)$, $\mathbf{k}_* = [k(\mathbf{x}_1, \mathbf{x}_*), \ldots, k(\mathbf{x}_{N_k}, \mathbf{x}_*)]^T$ and $k_{**} = k(\mathbf{x}_*, \mathbf{x}_*)$. The predictive distribution $p(R^*_{s\omega}|\mathcal{D}^k, \mathbf{x}_*) = \mathcal{N}(\mu_R, \sigma_R^2)$ is Gaussian with

$$\mu_R = \mathbf{k}_*^T (\mathbf{K} + \sigma_n^2 \mathbf{I})^{-1} \mathbf{R}_{s\omega}, \tag{4}$$

$$\sigma_R^2 = k_{**} - \mathbf{k}_*^T (\mathbf{K} + \sigma_n^2 \mathbf{I})^{-1} \mathbf{k}_*, \tag{5}$$

where $\mathbf{I}$ is the identity matrix. It is taken into consideration by the upper-level policy π_u to decide which context-option pair $(\mathbf{s}, k)$ to choose for each episode (line 5 and 6 in Alg. 1).

2) Outcome Reward Model: After each rollout, the quality of the actual trajectory $\boldsymbol{\tau}$ has to be evaluated. We learn a separate reward model $p(R_o|\mathcal{D}_{\succ}, \mathbf{o} = \phi(\boldsymbol{\tau}))$ from human pairwise preferences. Let $\mathcal{O} = \{\mathbf{o}_1, \ldots, \mathbf{o}_N\}$ be the outcomes of N rollouts across all K options, and $\mathcal{D}_{\succ} = \{\mathbf{v}_1 \succ \mathbf{u}_1, \ldots, \mathbf{v}_M \succ \mathbf{u}_M\}$ with $\mathbf{v}_j \in \mathcal{O}, \mathbf{u}_j \in \mathcal{O}$ be the learning set containing M pairwise preferences over outcomes. The outcomes are assumed to be valued according to a noisy utility function $y(\mathbf{o}) = R_o(\mathbf{o}) + \epsilon$, where R_o is the latent utility function of the human and $\epsilon \sim \mathcal{N}(0, \sigma_h^2)$ is i.i.d.

Gaussian noise. Using a probabilistic GP preference learning model [20], the latent rewards can be modeled as

$$R_o(\mathbf{o}) \sim \mathrm{GP}(m(\mathbf{o}), k(\mathbf{o}, \mathbf{o}')), \tag{6}$$

where the mean function $m(\cdot)$ is set to zero, and the squared exponential covariance function $k_{\mathrm{SE}}(\cdot,\cdot)$ is employed (Eq. 2).

In definite, we are looking for a function R_o such that $R_o(\mathbf{v}_i) > R_o(\mathbf{u}_i)$, for $i \in \{1 \dots M\}$. However, this does not yield a Gaussian likelihood function and hence the posterior distribution is not trivial to obtain. To compute the posterior we follow [20] and perform a Laplace approximation at the maximum a posteriori (MAP) estimate $\mathbf{r}_{\mathrm{MAP}} = [R_o(\mathbf{o}_1), \dots, R_o(\mathbf{o}_N)]|_{\mathbf{R}_o = \mathbf{r}_{\mathrm{MAP}}}$. The posterior $p(\mathbf{R}_o|\mathcal{D}_\succ)$ can be approximated as a Gaussian $\mathcal{N}(\mathbf{r}_{\mathrm{MAP}}, (\mathbf{K}^{-1} + \mathbf{\Gamma}_{\mathrm{MAP}})^{-1})$, where $\mathbf{\Gamma}_{\mathrm{MAP}} = \nabla^2 - \log p(\mathcal{D}_\succ|\mathbf{R}_o)|_{\mathbf{R}_o = \mathbf{r}_{\mathrm{MAP}}}$ is the Hessian of the negative log likelihood at the MAP estimate.

Therefore, the predictive distribution $p(\mathbf{R}_o^*|\mathcal{D}_\succ, \mathbf{r}, \mathbf{s})$ for two test instances $\mathbf{r} \in \mathcal{O}, \mathbf{s} \in \mathcal{O}$ with $\mathbf{R}_o^* = [R_o^*(\mathbf{r}), R_o^*(\mathbf{s})]^T$ becomes Gaussian as well, $\mathcal{N}(\bar{\mathbf{R}}_*, \mathbf{\Sigma}_*)$, with

$$\bar{\mathbf{R}}_* = [\bar{R}_r^*, \bar{R}_s^*]^T = \mathbf{K}_*^T \mathbf{K}^{-1} \mathbf{r}_{\mathrm{MAP}}, \tag{7}$$

$$\mathbf{\Sigma}_* = \begin{bmatrix} \Sigma_{rr}^* & \Sigma_{rs}^* \\ \Sigma_{sr}^* & \Sigma_{ss}^* \end{bmatrix} = \mathbf{K}_{**} - \mathbf{K}_*^T (\mathbf{K} + \mathbf{\Gamma}_{\mathrm{MAP}}^{-1})^{-1} \mathbf{K}_*, \tag{8}$$

where $\mathbf{K}$, $\mathbf{K}_*$ and $\mathbf{K}_{**}$ are defined analogously to Eq. 3. The predictive preference $p(\mathbf{r} \succ \mathbf{s}|\mathcal{D}_\succ)$ can be obtained by

$$p(\mathbf{r} \succ \mathbf{s}|\mathcal{D}_\succ) = \Phi\left(\frac{\mu_r^* - \mu_s^*}{\sigma_*}\right), \tag{9}$$

where $\Phi(\cdot)$ is the standard normal cumulative distribution function, and $\sigma_*^2 = 2\sigma_n^2 + \Sigma_{rr}^* + \Sigma_{ss}^* - \Sigma_{rs}^* - \Sigma_{sr}^*$.

We use the predictive distribution $p(R_o|\mathcal{D}_\succ, \mathbf{o})$ in two ways. First, to estimate the reward of outcome $\mathbf{o}$, which is added to dataset $\mathcal{D}^k = \{\mathbf{s}_i, \boldsymbol{\omega}_i, \mathbf{o}_i, \bar{R}(\mathbf{o}_i)\}_{i=1,\dots,N_k}$ of option k and used as a regression target for the training of the context-parameter reward model. Secondly, the predictive preference $p(\mathbf{r} \succ \mathbf{s}|\mathcal{D}_\succ)$ can be used to decide whether it is necessary to query the human for preference feedback. Since repeatedly giving feedback is a tedious task, it is desirable to minimize the number of queries. Thus, we only query the human if the expected information gained from a preference is larger than some threshold λ.

The information gain of including a preference $\mathbf{r} \succ \mathbf{s}$ can be expressed as the Kullback-Leibler (KL) divergence $\mathrm{KL}(p(\mathbf{R}_o|\mathcal{D}_{\succ+})||p(\mathbf{R}_o|\mathcal{D}_\succ))$ between the posterior after and before adding the preference, where $\mathcal{D}_{\succ+} = \mathcal{D}_\succ \cup \{\mathbf{r} \succ \mathbf{s}\}$ is an updated dataset that includes the new preference. Analogously, $\mathcal{D}_{\succ-}$ is defined by adding $\mathbf{s} \succ \mathbf{r}$ to $\mathcal{D}_\succ$. Weighting the information gain by the predictive preference in each of the two cases yields the following active learning criterion

$$\begin{aligned} &p(\mathbf{r} \succ \mathbf{s}|\mathcal{D}_\succ)\mathrm{KL}\left[p(\mathbf{R}_o|\mathcal{D}_{\succ+})||p(\mathbf{R}_o|\mathcal{D}_\succ)\right] \\ &\quad + p(\mathbf{s} \succ \mathbf{r}|\mathcal{D}_\succ)\mathrm{KL}\left[p(\mathbf{R}_o|\mathcal{D}_{\succ-})||p(\mathbf{R}_o|\mathcal{D}_\succ)\right] > \lambda, \end{aligned}$$

If the criterion is under the threshold λ, no human feedback is requested. The user-defined threshold parameter λ allows to trade off accuracy and data-efficiency, i.e. higher λ values lead to less feedback requests whereas lower λ values prompt many preferences but might lead to reward models with reduced bias.

B. Policy Learning

This section describes the policies used in the hierarchical framework. The upper-level policy π_u decides which context-option pair should be selected, whereas the lower-level policy $\pi_l^k(\boldsymbol{\omega}|\mathbf{s})$ of option k generates policy parameters $\boldsymbol{\omega}$.

1) Upper-Level Policy: Prior to each rollout, the robot has to decide which context-option pair $(\mathbf{s}, k)$ to choose in order to maximize the cumulative reward. The selection is performed by upper-level policy π_u using rewards $[\mu_R(\mathbf{s}, \boldsymbol{\omega}), \sigma_R^2(\mathbf{s}, \boldsymbol{\omega})]$. Having a probabilistic model of the reward function at hand, it is possible to use multi-armed bandits techniques as a way to trade off exploration and exploitation. We use the contextual GP-UCB acquisition function [33], [34]

$$\text{CGP-UCB}(\mathbf{s}, \boldsymbol{\omega}) = \mu_R(\mathbf{s}, \boldsymbol{\omega}) + \sqrt{\beta}\sigma_R(\mathbf{s}, \boldsymbol{\omega}), \tag{10}$$

where $\beta > 0$ is a trade-off parameter. To account for the uncertainty about the policy parameters $\boldsymbol{\omega}$, the expectation w.r.t. lower-level policy π_l^k needs to be considered, yielding

$$\pi_u : (\mathbf{s}^*, k^*) = \arg\max_{\mathbf{s},k} \mathbb{E}_{\boldsymbol{\omega} \sim \pi_l^k}\left[\text{CGP-UCB}(\mathbf{s}, \boldsymbol{\omega})\right] \tag{11}$$

$$\approx \arg\max_{\mathbf{s},k} \frac{1}{M} \sum_{j=1}^{M} \text{CGP-UCB}(\mathbf{s}, \boldsymbol{\omega}_j), \tag{12}$$

where the expectation of the acquisition function is approximated by samples $\boldsymbol{\omega}_j \sim \pi_l^k(\cdot|\mathbf{s})$.

2) Lower-Level Policy: Our goal is to learn K lower-level policies $\pi_l^k(\boldsymbol{\omega}|\mathbf{s})$, such that every policy maximizes the expected long-term reward given the observed data $\mathcal{D}^k = \{\mathbf{s}_i, \boldsymbol{\omega}_i, \mathbf{o}_i, \bar{R}(\mathbf{o}_i)\}_{i=1,\dots,N_k}$. As each policy $\pi_l^k(\boldsymbol{\omega}|\mathbf{s})$ depends on context $\mathbf{s}$, we employ contextual relative entropy policy search (REPS). Information-theoretic approaches like REPS attempt to stay close to the data while optimizing the policy. This behavior can be achieved by bounding the relative entropy or KL divergence between the context-parameter distribution $p(\mathbf{s}, \boldsymbol{\omega})$ and the observed joint distribution $q(\mathbf{s}, \boldsymbol{\omega})$. The whole constrained optimization problem is given by

$$\max_p \int p(\mathbf{s}, \boldsymbol{\omega}) R_{\mathbf{s}\boldsymbol{\omega}} d\boldsymbol{\omega} d\mathbf{s} \tag{13}$$

$$\text{s.t. } \mathrm{KL}(p(\mathbf{s}, \boldsymbol{\omega})||q(\mathbf{s}, \boldsymbol{\omega})) \leq \epsilon, \tag{14}$$

$$\int p(\mathbf{s}, \boldsymbol{\omega}) \boldsymbol{\psi}(\mathbf{s}) d\boldsymbol{\omega} d\mathbf{s} = \hat{\boldsymbol{\psi}}, \ \int p(\mathbf{s}, \boldsymbol{\omega}) d\boldsymbol{\omega} d\mathbf{s} = 1, \tag{15}$$

where ϵ is the desired KL bound and $\boldsymbol{\psi}(\mathbf{s}) \in \mathcal{R}^M$ are context features. We use the squared exponential function for defining the context features using M random context samples $\mathbf{s}_m$, i.e. the m-th element is defined as $\psi_m(\mathbf{s}) = k_{\mathrm{SE}}(\mathbf{s}, \mathbf{s}_m)$.

Eq. 15 ensures that the expected context features match the average context features $\hat{\psi}$ of the sample distribution. As shown in the original work [35], [36], it is possible to obtain a closed-form solution for this optimization problem using the method of Lagrangian multipliers,

$$p(\mathbf{s}, \boldsymbol{\omega}) \propto q(\boldsymbol{\omega}|\mathbf{s})\mu(\mathbf{s}) \exp\left(\frac{R_{\mathbf{s}\boldsymbol{\omega}} - \boldsymbol{\phi}(\mathbf{s})^T \boldsymbol{\nu}}{\eta}\right), \qquad (16)$$

where $\boldsymbol{\nu}$ and η are Lagrangian multipliers, and $\mu(\mathbf{s})$ is the context distribution. The lower-level policies are assumed to be Gaussian, $\pi_l^k(\boldsymbol{\omega}|\mathbf{s}) = \mathcal{N}(\boldsymbol{\omega}|\boldsymbol{\psi}(\mathbf{s})^T\mathbf{w}, \boldsymbol{\Sigma}_{\boldsymbol{\omega}})$, where $\mathbf{w}$ are linear weights on the context features $\boldsymbol{\psi}(\mathbf{s})$.

IV. EXPERIMENTS

In order to allow for reproducible evaluations, we synthesize preference feedback as follows. Given two subsequent outcomes $\mathbf{u}$ and $\mathbf{v}$, preference $\mathbf{u} \succ \mathbf{v}$ is generated if $R(\mathbf{u}) + \epsilon_u > R(\mathbf{v}) + \epsilon_v$, and $\mathbf{v} \succ \mathbf{u}$ otherwise. $\epsilon \sim \mathcal{N}(\epsilon|0, 1)$ is standard Gaussian noise and $R(\cdot)$ is a task-specific reward function.

A. Ball Throwing Task

The aim of this task is to learn how to throw a ball to a goal position $\mathbf{p}_{\text{goal}} \in \mathcal{R}^2$ on a hilly landscape given the horizontal coordinate of the goal position as context $s \in [2, 10]$. The simulated robot consists of three revolute joints of equal length. Depending on the context, the robot can decide between $K = 2$ options: the robot can throw the ball either from left $\left(\mathbf{p}_{\text{base}} = [-4, 0]^T\right)$ or right $\left(\mathbf{p}_{\text{base}} = [16, 0]^T\right)$ of the landscape. An example of a ball throw is depicted in Fig. 1. In this task the global and local contexts coincide. Hence, the task of the upper-level policy π_u reduces to selecting the best option k^* given context s. Each of the two options uses a separate lower-level policy $\pi_l^k(\boldsymbol{\omega}|s)$. The motion parameters $\boldsymbol{\omega} = (\mathbf{q}, \dot{\mathbf{q}}) \in \mathcal{R}^6$ consist of the goal joint angle and velocity of each of the three joints at ball release. The true reward R of a throw $\boldsymbol{\tau}$ with context s is given by

$$R(\boldsymbol{\tau}, s) = -\Delta\mathbf{p} - c_1 v_0^2 + c_2 \qquad (17)$$

where $\Delta\mathbf{p} = ||\mathbf{p}_{\text{goal}} - \mathbf{p}_{\text{hit}}||_2$ is the distance to the goal position, v_0 is the initial ball speed, and c_1 and c_2 are constants. The second term is a proxy for the required energy, such that in general higher rewards can be obtained by choosing the option that is closer to the context. Since above reward function is not provided to the learner, we learn an outcome reward model $p(R_o|\mathcal{D}_\succ, \mathbf{o})$ with $\mathbf{o} = (\Delta\mathbf{p}, v_0)$.

In the first experiment, we performed four trials with 500 rollouts. The reward models and policies were initialized from 25 random rollouts, where contexts for the left policy were uniformly sampled from the left side of the landscape, $s \sim U(2, 6)$, and contexts for the right policy were sampled from the right side, $s \sim U(6, 10)$. The policies and reward models are updated after every 5 and 50 rollouts, respectively. We compare our algorithm to two baselines:

- $K = 1$: We use the left policy for all rollouts, and provide contexts uniformly sampled from the landscape for initialization, i.e. $s \sim U(2, 10)$. This modification makes the use of a hierarchical architecture obsolete.
- no ORM: The outcome reward model (ORM) $p(R_o|\mathcal{D}_\succ, \mathbf{o})$ is omitted and the context-parameter reward models $p(R_{s\omega}|\mathcal{D}^k, s, \boldsymbol{\omega})$ are directly learned from preferences. This version of the algorithm is thus closely related to the algorithm introduced in [19] where the reward is learned from human preferences directly on the context-parameter space.

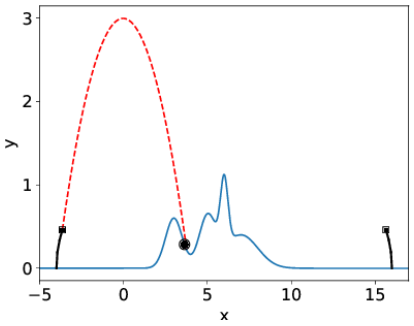
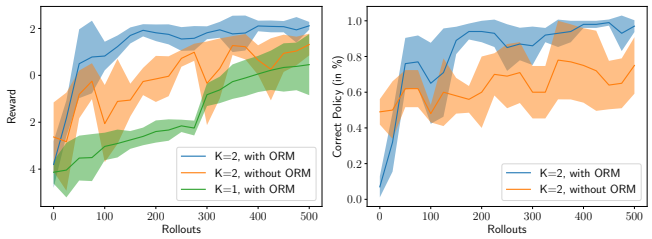


Fig. 1: Ball throwing example using the left policy $\left(\mathbf{p}_{\text{base}} = [-4, 0]^T\right)$. The red line depicts the trajectory of the ball after the end-effector (black square, left) has released the ball. The ball lands on the goal position $\mathbf{p}_{\text{goal}}$ (black circle).

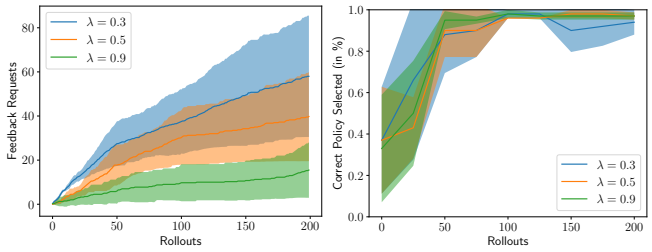
The results are shown in Fig. 2. Our approach combining the reward learning from both the robot and the human perspective achieves the best performance with a reward of 2.1 after 500 rollouts. It also shows a more stable learning behavior compared to the baselines as indicated by the faster convergence and smaller error bars. If no outcome reward model is used, the context-parameter reward models $p(R_{s\omega}|\mathcal{D}^k, s, \boldsymbol{\omega})$ have to be learned directly from preferences, which is much harder due to the higher dimensionality of the input space and uncertainty about the outcomes. Because errors in the reward estimation subsequently misguide the policy search, the performance of this variant is worse and suffers from high variance. As depicted in Fig. 2b, our approach chooses the correct policy 97% of the time after 500 rollouts. This percentage drops to 75% if no outcome reward model is used. The results suggest that learning multiple policies and a separate outcome reward model are important for efficiency.

In the next experiment, we evaluate the active learning component. We compare the number of requested queries as well as task performance using $\lambda = 0.3$, $\lambda = 0.5$ and $\lambda = 0.9$ for 200 rollouts in four trials. As shown in Fig. 3a, the number of feedback requests is reduced significantly as λ increases. For $\lambda = 0.3$ the amount of queries is cut to about 60, whereas a threshold of $\lambda = 0.9$ leads to a reduction to about 10 requests. It is worth noting that for all values of λ, less feedback is requested over time as the outcome reward model becomes more accurate. As depicted in Fig. 4, the large reduction of feedback requests for higher λ only leads to a moderate decrease in performance. Note that 3b shows that the variant with $\lambda = 0.9$ is even faster at selecting the correct option (i.e. to shoot with the arm closest to the goal)



(a) (b)

Fig. 2: Performance on ball throwing task compared to baselines. Our approach (blue) uses $K = 2$ options and a separate outcome reward model (ORM). The first baseline (yellow) omits the outcome reward model and directly learns the reward models on the context-parameter space as in [19]. The second baseline (green) learns only a single lower-level policy (no hierarchical policy). **(a)**: Average reward. Our proposed algorithm outperforms the baseline approaches and quickly surpasses a reward of 1.9. **(b)**: Average percentage of selecting the correct policy. Our approach is able to select the correct policy almost every time after 400 rollouts.



(a) (b)

Fig. 3: **(a)**: Average number of feedback requests for different values of λ. The higher λ, the less feedback was obtained. **(b)**: Average percentage of time the correct policy is selected for different values of λ. After 50 rollouts, the correct policy was chosen more than 90% of the time across all thresholds.

with less variance between the runs. We hypothesis that this is due to a faster convergence to a biased reward model that allows to choose the correct arm from which to shoot but does not result in the same precision of shots obtained with a smaller λ as depicted by the reward difference in Fig. 4.

B. Grasping Task

We apply our framework to learn pinch grasps, power grasps and medium wrap grasps [37] using the KUKA Light Weight Robot with a DLR/HIT II Hand. The motion parameters $\boldsymbol{\omega} = (\mathbf{p}_{\text{pre}}, \mathbf{p}_{\text{grasp}}, \mathbf{q}_{\text{grasp}}) \in \mathcal{R}^{10}$ consist of a pre-grasp and grasp position of the hand palm in task space, and the orientation during the grasp. The finger motion is fixed for every grasp type to reduce the number of parameters.

Grasp location candidates are generated by locally matching the point cloud of an object to contact parts collected from successful grasps. Given a database of M contact parts $\mathcal{C} = \{\mathbf{C}_1, \ldots, \mathbf{C}_M\}$, the Iterative Closest Points (ICP) algorithm [38] can be used to find the best match between the point cloud of an object and contact part $\mathbf{C}_j$.

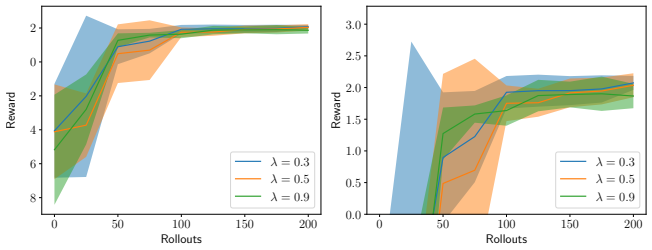
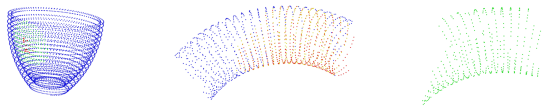


Fig. 4: Performance on ball throwing task for different active learning thresholds λ. **Left**: In all three cases, average rewards of 1.5 and above are reached after 100 rollouts. **Right** (magnified version): The variance in the rewards decreases most quickly for $\lambda = 0.9$, but the algorithm converges prematurely with a mean reward of 1.85. In contrast, mean reward values of about 2.05 are achieved with lower λ values.

Once an object point cloud $\mathbf{P}$ is obtained, we randomly sample a partial point cloud $\mathbf{p}_i \subset \mathbf{P}$. ICP iteratively estimates a homogeneous transformation matrix $\mathbf{H}_{ij}$, such that the transformed point cloud $\mathbf{H}_{ij}\mathbf{C}_j$ is most similar to $\mathbf{p}_i$ according to matching error d_{ij}. We choose the transformation matrix $\mathbf{H}_{ij^*}$ with the lowest error across all stored contact point clouds $\mathbf{C}_j \in \mathcal{C}$. A potential grasp part $\hat{\mathbf{p}}_i \subset \mathbf{P}$ on object point cloud $\mathbf{P}$ is then found in the vicinity of $\mathbf{H}_{ij^*}\mathbf{C}_{j^*}$. See Fig. 5 for results of the grasp part estimation using ICP. In order to reduce the dimensionality, we represent each grasp part $\hat{\mathbf{p}}_i$ by a grasp location vector $\mathbf{s}_i$ consisting of the center and normal vectors extracted from $\hat{\mathbf{p}}_i$ for the thumb and index finger. By repeating the procedure multiple times, a set of potential grasp locations is produced (the local contexts over which sub-policies of each grasp type are defined). This sampling procedure is applied to each grasp type separately, where each grasp type maintains its own set of contact parts. Since grasp locations are estimated locally, this approach is also feasible on a real system where a complete point cloud might not be available.



(a) (b) (c)

Fig. 5: Grasp part estimation. **(a)** Contact points (red) and contact part in the neighborhood of contact points $\mathbf{C}$ (green) on the point cloud of an object $\mathbf{P}$ (blue). If the contact led to a successful grasp, the contact part is added to dataset $\mathcal{C}$. **(b) - (c)** ICP result for partial point cloud $\mathbf{p}_i$ of an object (blue). The matched contact part $\mathbf{C_j}$ from dataset $\mathcal{C}$ is shown in red, whereas the best transformation of the ICP algorithm $\mathbf{H}_{ij^*}\mathbf{C_{j^*}}$ is highlighted in yellow. The estimated grasp part $\hat{\mathbf{p}}_i$ is shown in green.

We utilize the matching error d_{ICP} of each grasp location candidate $\mathbf{s}_i^k$ to guide the selection of a grasp type and location. The matching error is included directly into the

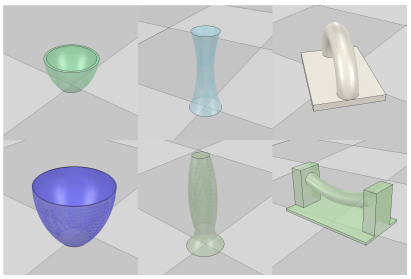
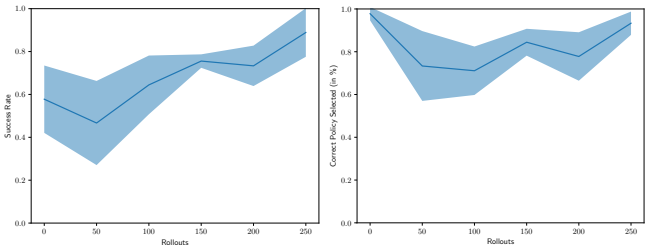


Fig. 6: Objects used for learning pinch grasps (left), power grasps (middle) and medium wrap grasps (right).



(a) (b)

Fig. 7: **(a)**: Average success rate (performance of the overall policy). **(b)**: Average percentage where the correct policy was selected (performance of the upper level policy). ICP prior provides a good initialization for the upper level policy. The learning of the reward models from preference feedback initially deteriorates the upper level policy, but after 250 rollouts, the reward models yield an upper level policy that recovers the performance of the ICP prior while they additionally allow for refinements of the lower level policy from 58% success rate to 89%. Plots are averaged over three runs.

acquisition function of upper-level policy π_u, yielding

$$\pi_u : (\mathbf{s}^*, k^*) = \arg\max_{\mathbf{s},k} \mathbb{E}[\text{CGP-UCB}(\mathbf{s}, \boldsymbol{\omega})] + \gamma d_{\text{ICP}}, \quad (18)$$

where $\gamma > 0$ is a scaling constant. To generate preferences from simulated grasps, we assume the following true reward function R for grasp $\boldsymbol{\tau}$ given object $\tilde{\mathbf{s}}$:

$$R(\boldsymbol{\tau}, \tilde{\mathbf{s}}) = c_1 \delta_{\text{lift}} - c_2 \Delta\mathbf{q} + \frac{1}{2} \log(\gamma_{\text{grasp}} \gamma_{\text{lift}}), \quad (19)$$

where $\Delta\mathbf{q} = ||\mathbf{q}_{\text{grasp}} - \mathbf{q}_{\text{lift}}||_2$ is the change in orientation of the object after it was lifted, γ_{grasp} and γ_{lift} are the number of contacts when the fingers are closed and the hand is subsequently lifted (with $\log(0) := 0$), and c_1 and c_2 are positive constants. $\delta_{\text{lift}} \in \{0, 1\}$ indicates whether the object was lifted at all. It is set to 1 if there are more than five contact points after lifting the object, $\gamma_{\text{lift}} > 5$, and 0 otherwise. The outcome features $\mathbf{o} = [\Delta\mathbf{p}_{xy}, \Delta p_z, \Delta\mathbf{q}]^T \in \mathcal{R}^3$ are the horizontal displacement, the vertical displacement and the change in orientation of the object after the grasp.

Throughout the experiments, each of the grasp types is initialized with 12 demonstrations, which are provided by fixing the motion parameters $\boldsymbol{\omega}$ by hand. For every subsequent rollout, an object point cloud is generated from which potential grasp locations are extracted per grasp type. Based on the estimated rewards, the robot selects a grasp type k^* and location $\mathbf{s}^*$, and generates the motion parameters $\boldsymbol{\omega}$ for grasping the object. Once the fingers are closed, we lift the arm and ask for feedback if necessary. Each lower-level policy π_l^k is updated after every 12 rollouts. To reduce the computational complexity, the reward models $p(R_o|\mathcal{D}_\succ, \mathbf{o})$ and $p(R_{s\omega}|\mathcal{D}^k, \mathbf{s}, \boldsymbol{\omega})$, including the hyper-parameter optimization step, is performed every three iterations, i.e. after every 36 rollouts.

In this experiment, the goal is to learn how to grasp the objects shown in Fig. 6. Each of the three grasp types is initialized with demonstrations of two of the objects. The ICP prior used in the upper level policy is a good prior explaining the high initial probability of selecting the correct grasp type (Fig. 7b). However, the lower level policies for each grasp type need to be optimized to improve the success rate of the overall policy (Fig. 7a).

As can be seen in Fig. 7, the upper policy is initially near optimal due to the strong prior given by the ICP while the reward model have initially zero mean. As the robot starts exploring and learning the reward models from preference feedback, the upper level policy initially deteriorates but it allows for the lower level policies to be refined. After 250 rollouts, the learned reward models allow the upper level policy to recover a similar success rate as when it was solely depending on the ICP prior while the refinements of the lower level policies allows to reach a significantly higher success rate (from 58% success rate to 89%).

V. CONCLUSION

We proposed in this paper a new framework for reinforcement learning from human feedback that aims at increasing both the sample and feedback efficiency so that the approach can be applied to physical systems with a human teacher. We demonstrated on a toy task that learning reward models from both the robot and the human perspectives can achieve the goal of improving on both efficiency metrics and constitutes an improvement over prior work that forced the reward model to be on a shared space. While we demonstrated on a grasping task how our reward learning approach can be incorporated into a hierarchical reinforcement algorithm to tackle more complex problems. In future work, larger scale experiments involving human teachers should be considered to assess the robustness of the reward models towards noise that might not be of Gaussian form. On a more technical note, one can also consider learning the outcome representation in an unsupervised way for tasks over which the trajectory compression function ϕ is not trivial to hand define.

REFERENCES

[1] A. Sahbani, S. El-Khoury, and P. Bidaud, "An overview of 3d object grasp synthesis algorithms," *Robot. Auton. Syst.*, vol. 60, no. 3, pp. 326–336, 2012.

[2] J. Bohg, A. Morales, T. Asfour, and D. Kragic, "Data-driven grasp synthesis – a survey," *IEEE Transactions on Robotics*, vol. 30, no. 2, pp. 289–309, 2014.

Preprint submitted to 2018 IEEE International Conference on Robotics and Automation. Received September 15, 2017.

[3] R. Balasubramanian, L. Xu, P. D. Brook, J. R. Smith, and Y. Matsuoka, "Physical human interactive guidance: Identifying grasping principles from human-planned grasps," *IEEE Transactions on Robotics*, vol. 28, no. 4, pp. 899–910, 2012.

[4] M. A. Roa and R. Suárez, "Grasp quality measures: review and performance," *Autonomous Robots*, vol. 38, no. 1, pp. 65–88, 2015.

[5] A. Y. Ng, D. Harada, and S. J. Russell, "Policy invariance under reward transformations: Theory and application to reward shaping," in *Proceedings of the Sixteenth International Conference on Machine Learning*, 1999, pp. 278–287.

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

[7] A. Y. Ng and S. J. Russell, "Algorithms for inverse reinforcement learning," in *Proceedings of the Seventeenth International Conference on Machine Learning*, 2000, pp. 663–670.

[8] C. Wirth, R. Akrour, G. Neumann, and J. Fürnkranz, "A survey of preference-based reinforcement learning methods," *Journal of Machine Learning Research*, 2017.

[9] D. C. Kingsley, "Preference uncertainty, preference refinement and paired comparison choice experiments," University of Colorado Boulder, Tech. Rep., 2006.

[10] L. L. Thurstone, "A law of comparative judgement," *Psychological Review*, vol. 34, pp. 278–286, 1927.

[11] R. Akrour, M. Schoenauer, and M. Sebag, *Preference-Based Policy Learning*, 2011, pp. 12–27.

[12] W. Cheng, J. Fürnkranz, E. Hüllermeier, and S.-H. Park, "Preference-based policy iteration: Leveraging preference learning for reinforcement learning," in *Machine Learning and Knowledge Discovery in Databases - European Conference, ECML PKDD 2011*, ser. Lecture Notes in Computer Science, D. Gunopulos, T. Hofmann, D. Malerba, and M. Vazirgiannis, Eds., vol. 6911. Springer Verlag, 2011, pp. 312–327.

[13] R. Akrour, M. Schoenauer, and M. Sebag, *APRIL: Active Preference Learning-Based Reinforcement Learning*, 2012, pp. 116–131.

[14] A. Wilson, A. Fern, and P. Tadepalli, "A Bayesian approach for policy learning from trajectory preference queries," in *NIPS*, 2012, pp. 1142–1150.

[15] R. Akrour, M. Schoenauer, J.-C. Souplet, and M. Sebag, "Programming by feedback," in *International Conference on Machine Learning (ICML)*, ser. ACM Int. Conf. Proc. Series, 2014.

[16] C. Wirth, J. Frnkranz, and G. Neumann, "Model-free preference-based reinforcement learning," in *Proceedings of the 30th AAAI Conference on Artificial Intelligence (AAAI-16)*, 2016, pp. 2222–2228. [Online]. Available: http://www.aaai.org/ocs/index.php/AAAI/AAAI16/paper/view/12247

[17] P. Christiano, J. Leike, T. B. Brown, M. Martic, S. Legg, and D. Amodei, "Deep reinforcement learning from human preferences," *arXiv preprint arXiv:1706.03741*, 2017.

[18] N. Thatte, H. Duan, and H. Geyer, "A sample-efficient black-box optimizer to train policies for human-in-the-loop systems with user preferences," *IEEE Robotics and Automation Letters*, vol. 2, no. 2, pp. 993–1000, 2017.

[19] A. Kupcsik, D. Hsu, and W. S. Lee, "Learning dynamic robot-to-human object handover from human feedback," in *Proceedings of the International Symposium on Robotics Research*, 2015.

[20] W. Chu and Z. Ghahramani, "Preference learning with gaussian processes," in *Proceedings of the 22nd International Conference on Machine Learning*, 2005, pp. 137–144.

[21] B. Shahriari, K. Swersky, Z. Wang, R. P. Adams, and N. de Freitas, "Taking the human out of the loop: A review of bayesian optimization," *Proceedings of the IEEE*, vol. 104, no. 1, pp. 148–175, 2016.

[22] 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.

[23] M. Cakmak and A. L. Thomaz, "Designing robot learners that ask good questions," in *Proceedings of the Seventh Annual ACM/IEEE International Conference on Human-Robot Interaction*, ser. HRI '12. New York, NY, USA: ACM, 2012, pp. 17–24.

[24] F. Stulp, E. Theodorou, M. Kalakrishnan, P. Pastor, L. Righetti, and S. Schaal, "Learning motion primitive goals for robust manipulation," in *International Conference on Intelligent Robots and Systems*, 2011, pp. 325–331.

[25] O. Kroemer, R. Detry, J. Piater, and J. Peters, "Combining active learning and reactive control for robot grasping," *Robotics and Autonomous Systems*, vol. 58, no. 9, pp. 1105–1116, 2010.

[26] T. Osa, J. Peters, and G. Neumann, "Experiments with hierarchical reinforcement learning of multiple grasping policies," in *Proceedings of the International Symposium on Experimental Robotics (ISER)*, 2016.

[27] M. P. Deisenroth, G. Neumann, and J. Peters, "A Survey on Policy Search for Robotics," *Foundations and Trends in Robotics*, pp. 388–403, 2013.

[28] A. Ijspeert and S. Schaal, "Learning Attractor Landscapes for Learning Motor Primitives," in *Advances in Neural Information Processing Systems (NIPS)*, ser. (NIPS). Cambridge, MA: MIT Press, 2003.

[29] R. Sutton, D. Precup, and S. Singh, "Between MDPs and Semi-MDPs: A Framework for Temporal Abstraction in Reinforcement Learning," *Artificial Intelligence*, vol. 112, pp. 181–211, 1999.

[30] A. G. Barto and S. Mahadevan, "Recent advances in hierarchical reinforcement learning," vol. 13, p. 2003, 2003.

[31] F. Stulp and S. Schaal, "Hierarchical Reinforcement Learning with Movement Primitives," in *2012 IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, 2011, pp. 231–238.

[32] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)*. The MIT Press, 2005.

[33] N. Srinivas, A. Krause, S. Kakade, and M. W. Seeger, "Gaussian process optimization in the bandit setting: No regret and experimental design," in *Proceedings of the 27th International Conference on Machine Learning*, 2010, pp. 1015–1022.

[34] A. Krause and C. S. Ong, "Contextual gaussian process bandit optimization," in *Proceedings of the 24th International Conference on Neural Information Processing Systems*, 2011, pp. 2447–2455.

[35] A. G. Kupcsik, M. P. Deisenroth, J. Peters, and G. Neumann, "Data-efficient generalization of robot skills with contextual policy search," in *Proceedings of the Twenty-Seventh AAAI Conference on Artificial Intelligence*, 2013, pp. 1401–1407.

[36] J. Peters, K. Mülling, and Y. Altun, "Relative entropy policy search," in *Proceedings of the Twenty-Fourth National Conference on Artificial Intelligence*, 2010, pp. 1607–1612.

[37] M. R. Cutkosky and R. D. Howe, *Human Grasp Choice and Robotic Grasp Analysis*, 1990, pp. 5–31.

[38] P. J. Besl and N. D. McKay, "A method for registration of 3-d shapes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, no. 2, pp. 239–256, 1992.