Learning Virtual Grasp with Failed Demonstrations via Bayesian Inverse Reinforcement Learning

Xu Xie^{*,1,2} Changyang Li^{*,1} Chi Zhang^{1,2} Yixin Zhu^{1,2} Song-Chun Zhu^{1,2}

***Abstract*— We propose Bayesian Inverse Reinforcement Learning with Failure (BIRLF), which makes use of failed demonstrations that were often ignored or filtered in previous methods due to the difficulties to incorporate them in addition to the successful ones. Specifically, we leverage halfspaces derived from policy optimality conditions to incorporate failed demonstrations under Bayesian Inverse Reinforcement Learning (BIRL) framework. Under the continuous control setting, the reward function and policy are learned in an alternative manner, both of which are estimated by function approximators to guarantee the learning ability. Our approach is formulated as a model-free Inverse Reinforcement Learning (IRL) method that naturally accommodates more complex environments with continuous state and action spaces. In experiments, we demonstrate the proposed method in a virtual grasping task, achieving a significant performance boost compared to existing methods.**

I. Introduction

Inverse Reinforcement Learning (IRL) [1] seeks to find the true reward function in a modified Markov Decision Process (MDP) [2] where the expert demonstrations are provided in place of a reward function. Intuitively, the task could be interpreted as learning a reward function that best explains a set of observed expert demonstrations. Nevertheless, no matter how a specific IRL problem is previously formulated, one key assumption prevails: the observed experts always execute (nearly) optimal actions induced from the underlying true reward function. Unfortunately, this strong assumption poses a strict constraint on the data collection stage: only successful demonstrations are recorded, and all failure cases are discarded. This hard constraint makes it extremely costly to collect demonstrations, particularly with a physical robot.

We argue, however, this strict requirement for IRL is unnecessary, not only because the failure cases are more readily available than the successful ones, but also because human beings are capable of learning with failure cases. Another aspect that we cannot ignore is that failed demonstrations, in terms of trajectory points made by experts, highly resemble successful ones; they usually differ only at a few critical points. Specifically, in the case of robot learning from object grasping demonstrations, the failed demonstrations are oftentimes distinctive from the successful ones in the very few last steps where the failed trajectories result in contact points that are slightly off stable grasps. In our observations, failed demonstrations are not made to be failed intentionally

* X. Xie, C. Li contributed equally to this work.
[1] UCLA Center for Vision, Cognition, Learning, and Autonomy. Emails: {xiexu, changyangli, chi.zhang, yixin.zhu}@ucla.edu, sczhu@stat.ucla.edu.
[2] International Center for AI and Robot Autonomy (CARA)

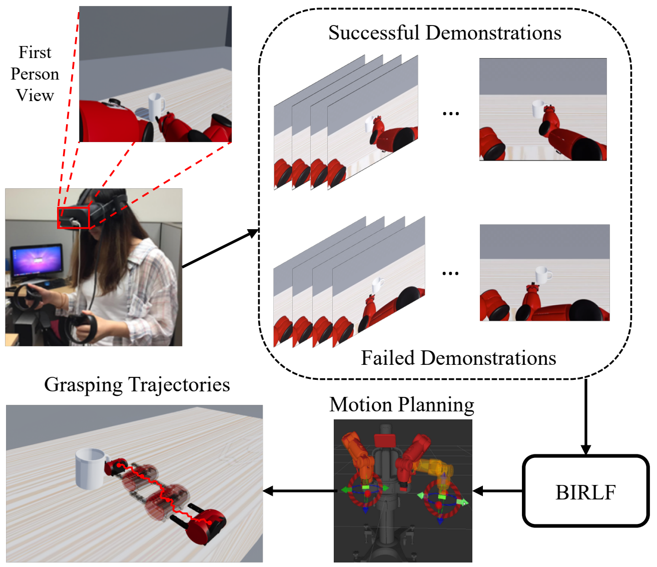


Fig. 1: Human demonstrators perform a complex grasping task that may include failed demonstrations in the virtual environment. The proposed BIRLF is able to infer the reward function from such a mix of both successful and failed demonstrations. The robot then learns the policy to achieve the demonstrated task, reproducing the grasping trajectories with a motion planner.

by the expert; both failed and successful demonstrations are initially pursuing the same trajectory distributions for one task. Motivated by such an observation, we propose an IRL method to incorporate failed demonstrations which could be informative to help with task learning.

Though Inverse Reinforcement Learning with Failure (IRLF) is not totally unprecedented, our proposed approach addresses several less-explored problems in IRLF. First, our approach naturally extends previous discrete state and action spaces [3] by accommodating a complex environment with continuous state and action spaces. Second, the proposed method automatically learns the feature function and composes the reward function by weights sampled from an Markov Chain Monte Carlo (MCMC) process, in contrast to previous [4] method using a set of hand-coded feature functions. Specifically, our method alternates between a weight sampling step by an MCMC sampler and a feature function learning step by policy iteration. Such a design allows us to leverage powerful function approximators to better estimate the true reward function. Third, as we will show in Section IV, from a Bayesian perspective, the proposed work could be interpreted as BIRLF with convex halfspace constraints, which, to the best of our knowledge, is the first attempt to cast IRLF in a Bayesian framework.

To evaluate the effectiveness of our approach, we develop a virtual environment where an autonomous agent is tasked to grasp various objects in a distance. By using this virtual environment, safe and successful expert demonstrations can be easily teleoperated and saved for training [5], [6]; see Figure 1 for the basic setup of the task. In each trial, the demonstrator was asked to control the virtual agent in a grasping task using Virtual Reality (VR) devices. Here, human subjects try to move the end-effector and configure the hand pose to a state that would most likely result in a stable grasp. The actual evaluation is done by a physics engine. In such a setting, both successful and failed trials could be automatically labeled and stored during demonstrations, thus greatly reducing efforts in the data collection process.

II. Related Work

Learning with Failed Demonstrations: In a typical setup of Learning from Demonstration (LfD) [7], *only* the expert demonstrations are given to improve the performance on robot manipulation tasks. Some works propose data efficient approaches [8], [9] to learn from less number of successful demonstrations, while failed demonstrations are widely regarded as noise [10], [11], [12], [13], [14], [15]. However, failed demonstrations should be leveraged as well since: (i) it is both time-consuming and demanding to collect only successful demonstrations, especially for complex tasks, and (ii) failed demonstrations also contain useful information which, when properly used, should improve performance.

Carefully designed algorithms using failed demonstrations have already shown notable performance improvement. Shiarlis *et al.* [4] introduce an early work on IRLF, where a new constraint is proposed in the maximum causal entropy framework [16] to encourage the learned policy to exhibit a different distribution from failed demonstrations, resulting in better convergence than its counterparts with successful demonstrations only. Lee *et al.* [3], [17] leveraged Gaussian process, in which a proficiency term is applied to generate both successful and failed demonstrations; this work showcases the potential effectiveness of adopting failed demonstrations in *GridWorld*. Learning from non-optimal demonstrations is also adopted by the Reinforcement Learning (RL) community; Gao *et al.* [18] learns an initial policy from both optimal and suboptimal demonstrations. However, these IRLF methods either rely on explicit feature representation or focus only on a low-dimensional state or action space, all of which are hardly effective in more complex IRLF tasks, such as the robot grasping one considered in this work.

Inverse Reinforcement Learning: *Only* expert demonstrations are usually provided in IRL [1], [19] to learn the unknown reward function in an MDP [2]. Existing methods could be divided into model-based and model-free methods [20], [21]; dynamics of a system are provided to the agent in model-based methods but not in model-free IRL. In this paper, we focus on model-free IRL.

Early works only require matched feature expectation between demonstrations and policies [22]. However, such a formulation is ill-posed; multiple, even an infinite number of, policies could produce the very same expectation, making the solution not unique. To find an optimal policy, Ratliff *et al.* [23] propose maximum margin planning to maximize the difference between other policies and the optimal one. BIRL [24] derives a posterior probability over the reward function space by combining prior knowledge and evidence from the experts' actions and demonstrates a significant improvement over heuristics-based methods. Later approaches apply the maximum entropy principle to find a distribution that matches the feature expectation [25]. As for variations of reward function's form, instead of modeling the reward function as a linear combination of features [22], Levine *et al.* [26] and Jin *et al.* [27] considers a nonlinear formulation using Gaussian processes; it determines the relevance of each feature with respect to the expert policy at the same time. Unfortunately, these methods consider only successful trials, leaving out many valuable resources during demonstrations, *i.e.*, failure cases accumulated at the data collection stage.

Robot Grasping: A robot grasp is defined as a task in which a target object is gripped by the fingers of a robot's end-effectors [28]. In general, grasp methods can be divided into two categories: analytical methods and empirical methods; see a recent review by Sahbani *et al.* [29]. Recently, there is an increasing interest in empirical methods to overcome the computational complexity posed by analytical methods. The majority of the recent works leverage deep RL for control policy learning [30], [31]. In our work, we propose a modified IRL method for robot grasping, with a reward function learned from human demonstrations.

III. Background and Notations

A. Markov Decision Process

The learning environment is characterized by an MDP defined by a tuple $\mathcal{M}=\{S,A,T,\gamma,r\}$, where S denotes the state space, A the action space, $T(s'|s,a)$ the transition probability from state s to s' by taking action a, γ the discount factor, and r the reward function. For LfD via IRL, the MDP tuple becomes $\mathcal{M}\setminus r$, but with trajectories $\mathcal{D}=\{d_1,\ldots,d_N\}$ from expert demonstrations. Each trajectory d_i is expressed as a sequence of state-action pairs.

B. Value Function of a Policy

A policy is defined as a mapping function from state space to action space: $\pi: S\mapsto A$. For a given policy π, the state value function $V^\pi(s)$ is defined as the expected accumulative reward when an agent starts from state s and executes π thereafter. Formally, the value function is expressed as:

$$V^\pi(s)=\mathbb{E}_{a\sim\pi}\left[\sum_{t=0}^{\infty}\gamma^t r(s_t)|s_0=s\right]. \tag{1}$$

A state-action value function Q could be similarly defined for each state-action pair (s,a):

$$Q^\pi(s,a)=r(s)+\gamma\mathbb{E}_{s'\sim T(\cdot|s,a)}[V^\pi(s')]. \tag{2}$$

Assuming optimality of the policy $\pi(s)$ under the reward function $r(s)$, the following inequalities should hold for all states and actions [19]:

$$\begin{gathered}\mathbb{E}_{s'\sim T(\cdot|s,\pi(s))}[V^\pi(s')]\geq\mathbb{E}_{s'\sim T(\cdot|s,a)}[V^\pi(s')]\\ Q^\pi(s,\pi(s))\geq Q^\pi(s,a)\\ \forall a\in A, s\in S.\end{gathered} \tag{3}$$

C. Bayesian Inverse Reinforcement Learning and PolicyWalk

BIRL [24] leverages probability distribution to represent the uncertainty in reward functions. Specifically, an agent (expert) $\mathcal{A}$ performs the task and generates a trajectory $d = \{(s_0, a_0), (s_1, a_1), ..., (s_T, a_T)\}$. Assuming $\mathcal{A}$ maximizes the accumulative reward and accomplishes the task using a stationary policy, the posterior probability of reward function r under Bayes theorem can be formulated as:

$$P(r|d) = \frac{P(d|r)P(r)}{P(d)} = \frac{1}{Z}\exp(\alpha \Sigma_t Q(s_t, a_t; r))P(r), \quad (4)$$

where Z is the normalizing constant, α the degree of confidence, and $Q(s_t, a_t; r)$ the state-action value function under the reward function r. However, estimating such a posterior is difficult in practice. To resolve this issue, PolicyWalk [24] was proposed as an efficient sampling procedure. When fixing the current reward function r, the sampler optimizes on-policy π to update Q function estimation. In the MCMC stage, the sampler samples $\tilde{r}$ from the neighborhood of the current r. After moving to the chain of $\tilde{r}$, the new optimal policy only requires several steps of policy iterations from the previous one. Ramachandran *et al.* [24] prove the rapid mixing property of the Markov chain with a uniform reward prior, providing a theoretical treatment of the convergence.

IV. Bayesian Inverse Reinforcement Learning with Failure

In this section, we detail the proposed method for learning with failed demonstrations. Our approach is formulated in a Bayesian framework by incorporating the value function inequalities in Equation 3, which implicitly define a valid convex set composed of halfspaces. In the following, we use D and F to denote the set of successful and failed demonstrations, respectively.

A. Problem Formulation

Starting from Equation 4, the posterior probability of the reward function r under the full demonstration set $\Psi = D \cup F$ could be expressed as:

$$P(r|\Psi) \propto P(\Psi|r)P(r). \quad (5)$$

To sample a reward function from the posterior, we use an MCMC chain so that the final reward function would not only match the statistics of trajectories in D but also capture the difference between D and F. With such an intuition, we decompose the likelihood from Equation 5 into:

$$P(\Psi|r) \propto \exp(\alpha \Sigma_{d \in \Psi, t} Q(s_{d,t}, a_{d,t}; r) + \beta \Delta U(D, F; r)), \quad (6)$$

where the first term follows Equation 4, $\Delta U(D, F; r)$ in the second term measures the potential difference between D and F, and β is a coefficient.

B. Halfspace-Induced Potential

To characterize $\Delta U(D, F; r)$ on different trajectory sets, we utilize Equation 3 and quantitatively compute this potential term among different solution spaces. Assume we parameterize the reward function $r(s)$ as $r(s) = \omega^T \phi(s)$,

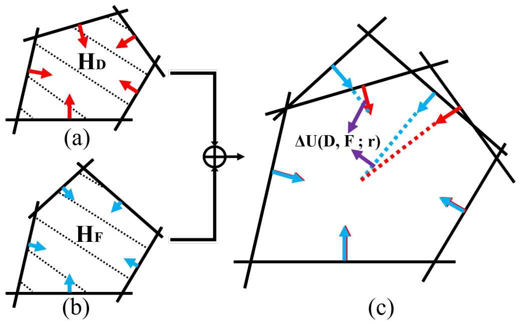


Fig. 2: Intersections of halfspaces (dashed area) with normal vectors (red and blue arrows) at boundaries for (a) successful demonstrations $\mathbb{H}_D$ and (b) failed demonstrations $\mathbb{H}_F$. (c) Measuring the potential difference among two demonstration sets.

where ω is the feature weights and $\phi(s)$ compactly represents the state features, the Q function could be reformulated as

$$Q(s, a; r) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t \omega^T \phi(s_t) | s_0 = s, a_0 = a\right] = \omega^T \mu(s, a), \quad (7)$$

where $\mu(s, a) = \mathbb{E}[\Sigma_{t=0}^{\infty} \gamma^t \phi(s_t) | s_0 = s, a_0 = a]$ is the feature expectation function. Substituting the Q function into Equation 3, we derive the following inequity:

$$\omega^T(\mu(s, \pi(s)) - \mu(s, a)) \geq 0, \forall s \in S, a \in A. \quad (8)$$

Solving ω in Equation 8 would give us an intersection of halfspaces. Following this observation, we define $\mathbb{H}_D$ as

$$\mathbb{H}_D : \{\omega | \omega^T(\mu(s_t, a_t) - \mu(s_t, a')) \geq 0, \forall s_t, a_t \in D, a' \in A\}, \quad (9)$$

and $\mathbb{H}_{D^\pi}$ on the current policy as

$$\mathbb{H}_{D^\pi} : \{\omega | \omega^T(\mu(s_t, \pi(s_t)) - \mu(s_t, a')) \geq 0, \forall s_t \in D, a' \in A\}. \quad (10)$$

Definitions of $\mathbb{H}_F$ and $\mathbb{H}_{F^\pi}$ follow in the same way.

We further note that each space defined above has a corresponding set of normal vectors with respect to its boundaries, denoted as N. As an example, the normal vector set of $\mathbb{H}_{D^\pi}$ is computed as

$$N_{\mathbb{H}_{D^\pi}} = \left\{ \frac{\mu(s_t, \pi(s_t)) - \mu(s_t, a')}{||\mu(s_t, \pi(s_t)) - \mu(s_t, a')||_2}, \forall s_t, a_t \in D, a' \in A \right\}. \quad (11)$$

Normal vector sets of $N_{\mathbb{H}_D}$, $N_{\mathbb{H}_F}$, and $N_{\mathbb{H}_{F^\pi}}$ are similarly obtained; see Figure 2 for a graphical illustration.

The potential term $\Delta U(D, F; r)$ thereby is defined as:

$$\Delta U(D, F; r) = \text{sim}(N_{\mathbb{H}_D}, N_{\mathbb{H}_{D^\pi}}) - \text{sim}(N_{\mathbb{H}_F}, N_{\mathbb{H}_{F^\pi}}), \quad (12)$$

where $N_{\mathbb{H}_D}$, $N_{\mathbb{H}_F}$, $N_{\mathbb{H}_{D^\pi}}$, and $N_{\mathbb{H}_{F^\pi}}$ are normal vector sets. The $\text{sim}(\cdot, \cdot)$ function over two sets could be generalized from one over two normal vectors. For example, if we define the vector similarity function as

$$\text{sim}(n_1, n_2) = (1 + \cos(n_1, n_2))/2, \quad (13)$$

the similarity over two sets can be computed by the following steps: (i) Get the most similar vector pair selected from two sets, respectively. (ii) Compute the similarity score and remove them from their own sets. (iii) Continue searching for the next vector pair, following step (i) and (ii), until one set is empty. (iv) The average of all similarity scores we obtain is treated as the similarity of two sets.

Algorithm 1: BIRLF algorithm

```
1: Collect demonstrations (e.g., trajectory sets for grasping) $D$ and $F$
2: Initialize network parameters $\theta$
3: Randomly initialize $\omega$ as an unit vector $||\omega||_2 = 1$
4: $\phi(s)$, $\mu(s,a)$, $\pi(a|s)$ := PolicyIteration($\theta$, $\omega$)
5: while not done do
6:     Randomly sample $\tilde{\omega}$ from the neighborhood of $\omega$
7:     Compute $\tilde{Q}_{D^{\pi}}(s_t, \pi(s_t)), \forall s_t, a_t \in D$ by Equation 7
8:     Compute $\tilde{Q}_{D}(s_t, a_t), \forall s_t, a_t \in D$ by Equation 7
9:     if $\exists s_t, a_t \in D$, $\tilde{Q}_{D^{\pi}}(s_t, a_t) < \tilde{Q}_{D}(s_t, a_t)$ then
10:        $\tilde{\phi}(s)$, $\tilde{\mu}(s,a)$, $\pi(a|s)$ := PolicyIteration($\theta$, $\tilde{\omega}$)
11:        Compute potential $\Delta U(D, F; r)$
12:        With probability $\min\{1, \frac{P(\tilde{r}|\Psi)}{P(r|\Psi)}\}$, $\omega := \tilde{\omega}$, $\pi := \tilde{\pi}$
13:    else
14:        With probability $\min\{1, \frac{P(\tilde{r}|\Psi)}{P(r|\Psi)}\}$, $\omega := \tilde{\omega}$
15:    end if
16: end while
17: Output $\theta$ and $\omega$
```

C. Algorithm

Under the proposed framework, we can leverage Equation 5 for MCMC sampling on the reward function using the PolicyWalk algorithm. This process is equivalent to sampling feature weights ω given a fixed $\phi(s)$. For feature function $\phi(s)$ and feature expectation function $\mu(s,a)$, we use function approximators to estimate their true values. Concretely, we instantiate both $\phi(s)$ and $\mu(s,a)$ as deep neural networks and learn them using Deep Deterministic Policy Gradient (DDPG) [32]. As shown in Figure 3, the network architecture is composed of four modules. $\phi(\cdot)$ takes state s as input and outputs the state feature. Actor module $A(\cdot)$ outputs an action distribution from $\phi(s)$. Different from the original DDPG, our network also outputs a vector as feature expectation $\mu(s,a)$, computed by a critic module $C(\cdot)$ that aggregates the action output from $A(\cdot)$ and contextual features derived from a mapping module $M(\cdot)$. The value function $Q(s,a)$ is then derived by an inner product of ω and $\mu(s,a)$. For fixed feature weights ω, network weights in each module are updated by policy gradient. Since ω is sampled from a Markov chain, the process of feature weight learning can be regarded as policy iteration. Note that the policy of the agent ($A(\phi(s))$) is updated in the meantime. We summarize our BIRLF algorithm in Algorithm 1.

D. Alternative

In practice, Equation 6 has a problem during the learning phase: at the early stage of training, $\Delta U(D,F;r)$ would fluctuate around zero due to the limited amount of data, leading to a biased posterior. To resolve this issue, we propose to incorporate an additional term to stabilize the posterior by ensuring the sum is above zero, *i.e.*,

$$P(\Psi|r) \propto \exp\left(\Sigma_{d\in\Psi,t} Q(s_{d,t}, a_{d,t}; r) + \beta_1 \Delta U(D,F;r) + \beta_2 |\Delta U(D,F;r)|\right), \quad (14)$$

where we append Equation 6 with the absolute potential $|\Delta U(D,F;r)|$. In our experiment described later in Section V, we initialize $\beta_2 \geq \beta_1$ and apply simulated annealing

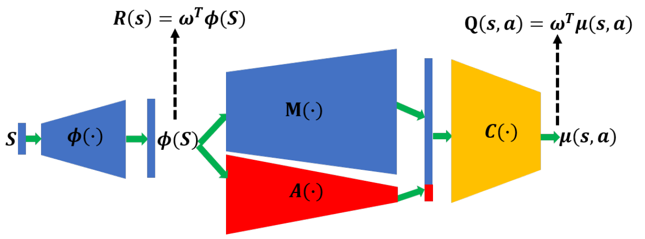


Fig. 3: The network architecture for the proposed BIRLF algorithm. $\phi(\cdot)$ is the state feature module, $M(\cdot)$ the mapping module, $A(\cdot)$ the actor module, and $C(\cdot)$ the critic module.

such that β_2 gradually approaches β_1. In the worst situation, when MCMC chain incorrectly fits F rather than D ($\Delta U(D,F;r) < 0$), the last two terms in Equation 14 will cancel each other ($\beta_1 = \beta_2$) and result in a posterior form exactly the same as BIRL in Equation 4. This alternative guarantees the performance of our method by incorporating the original BIRL sampling scheme as the baseline.

V. Experiments

A. Platform Setup

To properly measure the performance of different algorithms, a virtual environment with robot simulation [5], [33] is built for evaluation; see Figure 1. The virtual environment is built using Unreal Engine 4 (UE4) for its high rendering quality and real-time physics-based simulation. Behavior control of the virtual Baxter is implemented in the Robot Operating System (ROS), and the messages between UE4 and ROS are communicated on a customized UE4/ROS bridge using TCP/IP. Teleoperation is achieved using a set of Oculus Rift VR controllers.

The virtual scene for the grasping task includes a virtual Baxter and a target object to be grasped on top of a desk. The Baxter is initialized with the default untucked pose, and the object is initially placed at the center of the desk. The virtual grippers of the Baxter robot can be teleoperated to move around within a prescribed range of space, as long as the constraints of Baxter's joint limits are satisfied.

B. Task Design

In this virtual grasping task, the goal is to generate a stable grasp trajectory of a given object. Human demonstrators are put inside the virtual robot's embodiment with the robot's own joint limits and flexibility. Such a setup is designed to ease the transfer between different embodiments [34], but often leads to more failed human demonstrations.

C. Data Collection

Participants were first tasked with a training phase to be familiarized with the virtual world and the teleoperation. Using a pair of Oculus Rift controllers, participants were asked to control the gripper of the virtual Baxter robot from the robots' first-person view. During the data collection, the physics engine automatically evaluates whether a grasping attempt was successful. All grasping trajectories, including both the successful and failed ones, were recorded on-the-fly. In total, 120 demonstrations were collected for two types of objects: cuboids (87 successful and 33 failed) and mugs (76 successful and 44 failed); see examples in Figure 4.

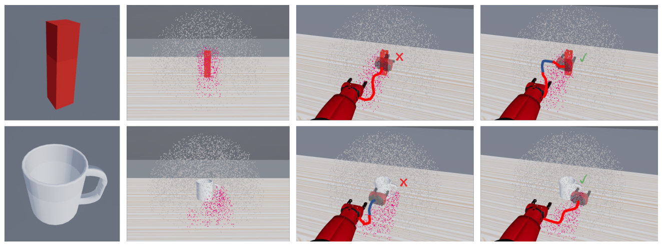


Fig. 4: 1st column: target objects: a cuboid and a mug. 2nd column: visualization of the set S_G (in red dots) and the set $S_{D \setminus G}$ (in gray dots); see text for details. 3rd column: failure cases. 4th column: success cases. Blue trajectories gain less rewards than the red ones.

D. Training

In the training stage, the virtual Baxter is spawned in the same location in the virtual scene as in the data collection phase. An additional IRL module is inserted between the virtual environment in UE4 and the motion planner in ROS (see Figure 1) to guide robot grasping autonomously.

The state space of the environment is $\{s_w, s_{t_1}, s_{t_2}\} = \{(x_w, y_w, z_w), (x_{t_1}, y_{t_1}, z_{t_1}), (x_{t_2}, y_{t_2}, z_{t_2})\}$, a 9-D vector with w, t_1 and t_2 representing gripper *wrist*, *left tip end*, and *right tip end*, respectively. We parameterize the action space using a 4-D vector $(a_x, a_y, a_z, a_t) \in [-1.0, 1.0]^4$: the first three fields are the *normalized* movement along each axis of the 3D Cartesian plane, and the last field a_t is the 1-D movement of two finger tips of the gripper.

Within one training episode, the robot sends its state s as a "state" message to the IRL module (UE4 to IRL). By estimating the reward value as well as the current policy, the IRL agent generates an action a and sends it as an "action" message (IRL to ROS). After the "action" message is received, the virtual robot gripper executes the action through motion planning and updates the state to s' (ROS to UE4). This process repeats itself until termination.

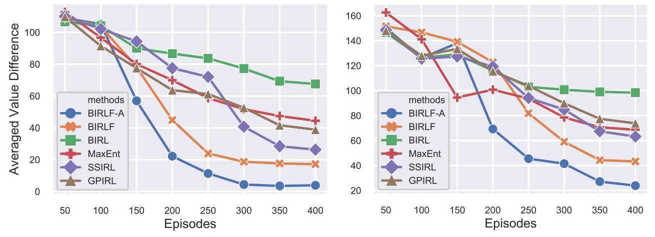
One episode will terminate when one of the following four scenarios happens: (i) a successful grasp, (ii) the target object falls down, (iii) any part of the robot touches things other than the target object, or (iv) the length of the action sequence exceeds the maximum number allowed (100 steps). A grasp validation routine will be called to determine whether the grasp is successful.

E. Experimental Results

We now detail virtual grasping experiments using both cuboids and mugs. Experiments on more objects can be found in the supplementary video. We define the ground truth reward function $r(s)$ in the virtual environment to fairly compare different IRL methods

$$r(s) = \begin{cases} 0.02, & s \in \Omega_G \\ 0.01, & s \in \Omega_{D \setminus G} \\ -0.01, & \text{otherwise}, \end{cases} \tag{15}$$

where Ω_G represents the convex hull formed by the success frames in the demonstrations, *i.e.*, the set S_G of $\{s_{i,T_i} : \forall (s_{i,T_i}, a_{i,T_i}) \in D\}$ (red dots in Figure 4), and $\Omega_{D \setminus G}$ the convex hull of the set $S_{D \setminus G}$ of $\{s_{i,j} : \forall (s_{i,j}, a_{i,j}) \in D, \forall j \neq T_i\}$ (gray dots in Figure 4). This design of the ground truth reward function will incur higher rewards for the red trajectories in Figure 4 and lower ones for blue segments.



(a) Cubiods (b) Mugs

Fig. 5: Averaged value difference (AVD) for different IRL methods.

We evaluate the proposed IRLF algorithm by comparing against the following state-of-the-art IRL methods:

- **BIRL [24].** Bayesian-IRL handles the uncertainty of reward estimation using a probability distribution. See Section III and [24] for details.
- **MaxEntIRL [25].** Maximum Entropy-IRL is also a probabilistic approach to reason about uncertainty in reward function, providing a well-defined, globally normalized distribution over decision sequences by matching feature expectation. We implement it by empirically estimating state visitation frequencies from sampled trajectories.
- **SSIRL [35].** Semi-supervised-IRL relaxes the conditions of IRL by not requiring all the demonstrations to be generated by an expert, *i.e.*, failure cases are allowed. This method is inspired by semi-supervised SVM [36], and treats feature expectations as the labels and optimizes the objective using a minimax scheme. We implement the feature expectations by leveraging sample trajectories.
- **GPIRL [26].** Gaussian Process IRL learns the reward as a nonlinear function. This function is approximated through Gaussian process, and its structure is determined by the kernel function. We empirically estimate state and action visitation frequencies from sampled trajectories in our implementation.

We denote the proposed IRLF method using Equation 6 as BIRLF and the one using Equation 14 as BIRLF-A. For BIRLF, we set $\alpha = 1.0$, $\beta = 10^2$. For BIRLF-A, we set $\alpha = 1.0$, $\beta_1 = 10^2$, and $\beta_2 = 10^3$, and applied annealing every 50 episodes to linearly decrease β_2 from 10^3 to 10^2.

We adopt the following two metrics to measure the performance of the methods:

- **Average Value Difference (AVD).** This metric represents the difference of the average returns between successful demonstrations by human participants and the ones generated by an IRL algorithm.
- **Mean Squared Error (MSE).** We compute the MSE of relative reward difference, calculated between each time frame of trajectories from the successful human demonstrations and the ones generated by an IRL agent.

Figure 5 compares the results of the proposed method BIRLF with other baselines. For methods that take failed demonstrations into account (BIRLF and SSIRL), we create a shared demonstration set with $|D| = 2|F|$. For methods that only assume expert demonstrations (BIRL, MaxEntIRL, and GPIRL), we create demonstration sets consisting only of

TABLE I: Quantitative evaluation (MSE) of different IRL methods on grasping cuboids and mugs. The performance is measured across different ratio configurations $|D|:|F|$.

Methods	$|D|:|F|$					
	$1:1$		$2:1$		$4:1$	
	Cuboid	Mug	Cuboid	Mug	Cuboid	Mug
BIRLF-A	**0.90**%	**3.95**%	**0.72**%	**2.44**%	0.84%	**2.53**%
BIRLF	1.30%	5.88%	0.78%	3.12%	**0.82**%	3.93%
BIRL	8.00%	9.67%	7.15%	8.42%	7.20%	9.09%
SSIRL	2.17%	5.98%	1.15%	5.19%	1.18%	5.26%
MaxEntIRL	4.33%	6.10%	3.99%	5.77%	4.13%	6.04%
GPIRL	4.81%	7.20%	3.65%	6.29%	3.79%	6.84%

successful demonstrations. For fair comparisons, we keep the sizes of demonstration sets the same for different methods. As shown in Figure 5, our proposed methods (BIRLF and BIRLF-A) converge after 400 training episodes and achieve the minimum AVD.

We show the MSE of different methods on both cuboids and mugs in Table I. The performance of different methods was evaluated after 600 episodes of training when all methods had reached convergence. We follow a common practice to present the performance of our method under different size ratios between the successful demonstration set and the failed demonstration set $|D|:|F|$, while keeping the size of the full demonstration set Ψ fixed. Note that for BIRL and MaxEntIRL, methods that only handle successful demonstrations, only D is provided for training.

As shown in the tables, the proposed approaches, BIRLF and BIRLF-A, generally achieve the lowest MSE compared to others. We also notice BIRLF-A performs better than BIRLF in general; the only exception occurs when the algorithm learns to grasp the mug with $|D|:|F|=4:1$.

VI. Conclusion

In this paper, we propose Bayesian IRLF that incorporates both successful and failed demonstrations for continuous state/action space. Halfspaces from policy optimality are leveraged to model the failed demonstrations set. In a complex virtual grasping task, our method achieves state-of-the-art performance compared to four existing methods.

Acknowledgement: The authors thank Prof. Ying Nian Wu, Feng Gao, Mark Edmonds from UCLA for useful discussions. The work reported herein was supported by DARPA XAI grant N66001-17-2-4029, ONR MURI grant N00014-16-1-2007, ARO grant W911NF-18-1-0296, and an NVIDIA GPU donation grant.

References

[1] S. Russell, "Learning agents for uncertain environments," in *ICML*, 1998.
[2] M. L. Puterman, *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.
[3] K. Lee, S. Choi, and S. Oh, "Inverse reinforcement learning with leveraged gaussian processes," in *IROS*, 2016.
[4] K. Shiarlis, J. Messias, and S. Whiteson, "Inverse reinforcement learning from failure," in *AAMAS*, 2016.
[5] X. Xie, H. Liu, Z. Zhang, Y. Qiu, F. Gao, S. Qi, Y. Zhu, and S.-C. Zhu, "Vrgym: A virtual testbed for physical and interactive ai," *arXiv preprint arXiv:1904.01698*, 2019.
[6] T. Zhang, Z. McCarthy, O. Jow, D. Lee, K. Goldberg, and P. Abbeel, "Deep imitation learning for complex manipulation tasks from virtual reality teleoperation," *arXiv preprint arXiv:1710.04615*, 2017.
[7] B. D. Argall, S. Chernova, M. Veloso, and B. Browning, "A survey of robot learning from demonstration," *Robotics and autonomous systems*, vol. 57, no. 5, pp. 469–483, 2009.
[8] P. Englert and M. Toussaint, "Combined optimization and reinforcement learning for manipulation skills.," in *RSS*, 2016.
[9] K. Chatzilygeroudis, R. Rama, R. Kaushik, D. Goepp, V. Vassiliades, and J.-B. Mouret, "Black-box data-efficient policy search for robotics," in *IROS*, 2017.
[10] A. Coates, P. Abbeel, and A. Y. Ng, "Learning for control from multiple demonstrations," in *ICML*, 2008.
[11] D. Silver, J. Bagnell, and A. Stentz, "High performance outdoor navigation from overhead data using imitation learning," *RSS*, 2008.
[12] A. Coates, P. Abbeel, and A. Y. Ng, "Apprenticeship learning for helicopter control," *Communications of the ACM*, vol. 52, no. 7, pp. 97–105, 2009.
[13] N. D. Ratliff, D. Silver, and J. A. Bagnell, "Learning to search: Functional gradient techniques for imitation learning," *Autonomous Robots*, vol. 27, no. 1, pp. 25–53, 2009.
[14] F. S. Melo, M. Lopes, and R. Ferreira, "Analysis of inverse reinforcement learning with perturbed demonstrations.," in *ECAI*, 2010.
[15] J. Zheng, S. Liu, and L. M. Ni, "Robust bayesian inverse reinforcement learning with sparse behavior noise.," in *AAAI*, 2014.
[16] B. D. Ziebart, *Modeling purposeful adaptive behavior with the principle of maximum causal entropy*. PhD thesis, CMU, 2010.
[17] S. Choi, K. Lee, and S. Oh, "Robust learning from demonstration using leveraged gaussian processes and sparse-constrained optimization," in *ICRA*, 2016.
[18] Y. Gao, J. Lin, F. Yu, S. Levine, T. Darrell, *et al.*, "Reinforcement learning from imperfect demonstrations," *arXiv preprint arXiv:1802.05313*, 2018.
[19] A. Y. Ng, S. J. Russell, *et al.*, "Algorithms for inverse reinforcement learning.," in *ICML*, 2000.
[20] T. Osa, J. Pajarinen, G. Neumann, J. A. Bagnell, P. Abbeel, J. Peters, *et al.*, "An algorithmic perspective on imitation learning," *Foundations and Trends® in Robotics*, vol. 7, no. 1-2, pp. 1–179, 2018.
[21] S. Arora and P. Doshi, "A survey of inverse reinforcement learning: Challenges, methods and progress," *arXiv preprint arXiv:1806.06877*, 2018.
[22] P. Abbeel and A. Y. Ng, "Apprenticeship learning via inverse reinforcement learning," in *ICML*, 2004.
[23] N. D. Ratliff, J. A. Bagnell, and M. A. Zinkevich, "Maximum margin planning," in *ICML*, 2006.
[24] D. Ramachandran and E. Amir, "Bayesian inverse reinforcement learning," *Urbana*, vol. 51, no. 61801, pp. 1–4, 2007.
[25] B. D. Ziebart, A. L. Maas, J. A. Bagnell, and A. K. Dey, "Maximum entropy inverse reinforcement learning.," in *AAAI*, 2008.
[26] S. Levine, Z. Popovic, and V. Koltun, "Nonlinear inverse reinforcement learning with gaussian processes," in *NIPS*, 2011.
[27] M. Jin, A. Damianou, P. Abbeel, and C. Spanos, "Inverse reinforcement learning via deep gaussian process," *arXiv preprint arXiv:1512.08065*, 2015.
[28] K. B. Shimoga, "Robot grasp synthesis algorithms: A survey," *IJRR*, vol. 15, no. 3, pp. 230–266, 1996.
[29] A. Sahbani, S. El-Khoury, and P. Bidaud, "An overview of 3d object grasp synthesis algorithms," *Robotics and Autonomous Systems*, vol. 60, no. 3, pp. 326–336, 2012.
[30] F. Zhang, J. Leitner, M. Milford, B. Upcroft, and P. Corke, "Towards vision-based deep reinforcement learning for robotic motion control," *arXiv preprint arXiv:1511.03791*, 2015.
[31] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, p. 529, 2015.
[32] Y. Duan, X. Chen, R. Houthooft, J. Schulman, and P. Abbeel, "Benchmarking deep reinforcement learning for continuous control," in *ICML*, 2016.
[33] H. Liu, Z. Zhang, X. Xie, Y. Zhu, Y. Liu, Y. Wang, and S.-C. Zhu, "High-fidelity grasping in virtual reality using a glove-based system," in *ICRA*, 2019.
[34] H. Liu, C. Zhang, Y. Zhu, C. Jiang, and S.-C. Zhu, "Mirroring without overimitation: Learning functionally equivalent manipulation actions," in *AAAI*, 2019.
[35] M. Valko, M. Ghavamzadeh, and A. Lazaric, "Semi-supervised apprenticeship learning," in *European Workshop on Reinforcement Learning*, 2013.
[36] K. P. Bennett and A. Demiriz, "Semi-supervised support vector machines," in *NIPS*, 1999.