Learn to Outperform Demonstrators via Reward and Policy Co-learning

Mingkang Wu
mingkang.wu@utsa.edu
Department of Electrical and Computer Engineering
The University of Texas at San Antonio

Feng Tao
feng.tao2@us.bosch.com
Robert Bosch LLC in Sunnyvale, CA 94085

Yongcan Cao
yongcan.cao@utsa.edu
Department of Electrical and Computer Engineering
The University of Texas at San Antonio

Abstract

In this work, we study the problem of obtaining a better control policy from demonstrations in Markov decision processes where both the environment dynamics and the reward function are unknown to the learning agent. The most relevant solution is the inverse reinforcement learning (IRL). The performance of the obtained control policy from IRL, however, heavily depends on the quality of demonstrations and hardly outperforms them. To overcome this limitation, we propose a novel method that enables the learning agent to outperform the demonstrator via a co-learning strategy with a general stereo utility. In particular, we propose a new stereo utility definition that aims to address the bias in the interpretation of good demonstrations via providing a hybrid view of future rewards under different discounting factors. We then propose a new loss function for the learning agent to co-learn the reward function and control policies such that the learning agent can outperform the demonstrator. The performance of the proposed algorithm is further validated in three OpenAI environments and a real-world indoor drone flight scenario.

1 Introduction

Reinforcement learning (RL) has shown its advantages in yielding human-level or better-than-human-level performance in, *e.g.*, Go and Atari games (Mnih et al., 2015; Silver et al., 2016). The basic idea of RL is to learn control policies that optimize certain metrics. For example, many existing RL algorithms, such as DQN (Mnih et al., 2015), DDPG (Lillicrap et al., 2015) and proximal policy optimization (PPO) (Schulman et al., 2017), leverage the cumulative reward as the metric to evaluate the performance of a control policy.

In practice, however, the reward function itself may not be available or optimal. Literature, such as (Ng et al., 1999; Wu et al., 2023) have proved that applying techniques like reward shaping can be beneficial to the policy search process. For example, the authors in (Kumar et al., 2018) observed that a properly shaped reward function can yield a faster walking agent than that trained with the original reward function, indicating that the original reward function may not be optimal. Moreover, the given reward function may suffer from corruption caused by over-complicated reward settings (Everitt et al., 2017). The noisy reward settings can consequently mislead the learning agents to some degree (Huang & Zhu, 2019). To overcome these challenges, it is important to learn task-specific reward functions for the design of optimal control policies that best fulfill the task objectives.

One important and popular approach for the learning/approximation of the reward function is the inverse reinforcement learning (IRL) (Ng et al., 2000; Ziebart et al., 2008; Choi & Kim, 2011; Finn et al., 2016; Šošić et al., 2018; Hu et al., 2020; Jiang et al., 2021; Devidze et al., 2022), which seeks to learn the reward function from expert demonstrations. The traditional IRL methods, however, have two main limitations. First, they follow the common template for IRL (Arora & Doshi, 2018), *i.e.*, optimize the policy in the inner loop of reward learning given unknown dynamics. As the policy optimization procedure can be very complex, it is challenging for the traditional IRL methods to find an optimal policy in high-dimensional systems. A sample-based approach (Finn et al., 2016) was proposed to address this challenge. However, the approach is based on policy optimization with local linear models (Levine & Abbeel, 2014) and does not allow for learning from task rewards, which lacks specific guidance. The second limitation of the traditional IRL methods is that the performance of IRL is typically dominated by the performance of the demonstrator. There are limited existing works addressing the problem of achieving better-than-demonstrator performance (Brown et al., 2020), such as the maximum entropy IRL (Ziebart et al., 2008; Wu et al., 2020), that address the noise in the demonstrations and could result in a better policy than the demonstrator, while requiring that most demonstrations come from an expert. Note also that the proposed methods in (Ziebart et al., 2008; Wu et al., 2020) do not focus on learning to outperform demonstrations directly, which can not guarantee obtaining better policies. Moreover, The approach proposed in (Kang et al., 2018) aims to guide the exploration by augmenting the sparse reward signals through measuring the divergence between RL current policy and a provided expert policy. The work in (Nair et al., 2018) combines RL with imitation learning to avoid the exploration issue in RL, which could yield policies that outperform the demonstrators. The authors in (Brown et al., 2019; 2020) propose to use a set of ranked demonstrations to learn a reward function that allows better-than-demonstrator performance. Similarly, a rating-based RL method proposed in (White et al., 2024) is designed to infer a reward function based on human ratings data, enabling effective policies learning. However, the existing approaches either lack the recovery of the intention/reward of the demonstrator, which is reusable for RL, or are inefficient because the task is solved into two sequential steps, namely, recovering the reward function and extracting a policy from the derived reward function with RL.

Another relevant work is on the use of non-exponential discounting in RL. In (Fedus et al., 2019), the authors proposed a new hyperbolic discounting mechanism in RL and found that value functions learned over multiple time-horizons can help obtain better performance over some state-of-the-art value-based RL methods. This is an interesting direction towards performance improvement via using multiple time-horizons (discount factors). However, no existing work has been studied on IRL with multiple discount factors, which is one key motivation of our current work.

In this paper, we propose a new reward and policy concurrent learning approach to recover the reward function and derive control policies that can mimic and then outperform the demonstrator. We first propose a stereo utility definition that calculates the expectation of the utility value for one trajectory/demonstration with respect to different discount factors. We then propose a new loss function aimed at enabling the learning agent to exceed the demonstrator. The loss function includes both the policy learning process and the reward function approximation process, hence yielding a concurrent learning structure. Lastly, we demonstrate the effectiveness of our proposed algorithm in both virtual and real-world environments.

2 Preliminaries and Problem Statement

A standard Markov decision process (MDP) can be represented as a tuple given by $\mathcal{M} := \langle S, A, T, R, \gamma \rangle$, where S denotes the state space, A the action space, T the transition model, R the reward function, and $\gamma \in [0, 1)$ is the discount factor that is used to compute a weighted accumulation of past rewards for a trajectory $\{s_0, a_0, r_0, ..., s_j, a_j, r_j, ...\}$, where $j \in \mathbb{N}$, $s_j \in S$, and $a_j \in A$. The selection of an action a_j is determined by the action policy represented as $\pi_\theta(a_j|s_j) : S \to p(A|S, \theta)$, where θ is the parameter of the action policy generated, *e.g.*, by a neural network. Note that $p(\cdot)$ can be either stochastic or deterministic. r_j is an immediate reward for state s_j after taking the action a_j, which is derived from the reward function R. In standard RL, the goal for the RL agent is

to learn a policy that can maximize the (discounted) accumulated reward (Sutton & Barto, 2018). Since R is typically assumed to be available, the (discounted) cumulative reward is widely used in the existing RL approaches as the metric to evaluate the policy. When R is unavailable, a learned reward function is required when using the existing RL approaches.

Here we consider a RL-MDP$\backslash R$ problem where agent has a specific goal, while R is unavailable. Note that the RL-MDP$\backslash R$ is different from MDP$\backslash R$ by including the simultaneous learning of action policies that aim to outperform the demonstrator. We adopt the reward function specification (Arora & Doshi, 2018) as $R : S \to \mathbb{R}$, which provides a map from the state to the associated immediate reward (Wulfmeier et al., 2015). To approximate the unknown relationship between states and rewards, a neural network architecture $g : \mathbb{R}^n \to \mathbb{R}$ can be used to take raw state features $s_j \in S$, which could be high dimensional, as the input and return an immediate reward value. The approximated reward function $\hat{R}$ is then represented as

$$\hat{R} := g(x|\phi), \tag{1}$$

where $x \in S$ and ϕ is the parameter associated with the neural network. We denote one sampled trajectory with an initial state s_0 in the RL-MDP$\backslash R$ setting as $\tau^{\theta^+}_{s_0}$. The trajectory $\tau^{\theta^+}_{s_0}$ is sampled from the learner's policy π_{θ^+} (can be either stochastic or deterministic) governed by parameters θ^+, such that $\tau^{\theta^+}_{s_0} = \{s_0, \pi_{\theta^+}(a_0|s_0), s_1, ..., \pi_{\theta^+}(a_{T-1}|s_{T-1}), s_T\}$, where T represents the length of the sampled trajectory. Similarly, we denote the demonstration with an initial state s_0 as $\tau^{\theta^*}_{s_0} = \{s_0, (a_0), s_1, (a_1)..., s_{T^*-1}, (a_{T^*-1}), s_{T^*}\}$, where T^* represents the length of the episode associated with that demonstration and the parentheses indicate that the action information is not required to be available (since $\hat{R}$ defined in (1) only considers the raw state information as the features). If the initial state and the policy are not given specifically, the demonstration/trajectory notation can be simplified as τ^θ. The discounted cumulative reward of one demonstration/trajectory τ^θ is given by $G_\phi(\tau^\theta, \gamma) = \sum_{j=0}^{T} \gamma^j g(s_j|\phi)$, where $s_j \in \tau^\theta$, and the neural network output value $g(s_j|\phi)$ is an estimation of the corresponding immediate reward r_j.

Note that the ground truth reward function R is unavailable for comparison. Hence, our goal is to obtain learned reward functions $\hat{R}$ such that the policy learning process converges to a final policy that yields exceeding and stable performance, and $\hat{R}$ can quantitatively distinguish trajectories, *i.e.*, good trajectories yield larger discounted cumulative rewards than the bad ones.

3 Reward and Policy Co-Learning

The existing research on IRL usually chooses one single discount factor γ for calculating the discounted cumulative rewards with respective to the expert's demonstrations. For example, the studies in (Shiarlis et al., 2016) define $\gamma = 1$ for the finite length trajectories, with an inherent assumption that the expert takes the rewards accumulated in one trajectory equally. However, the expert's principle on viewing future rewards, *i.e.*, γ, may not be static at different future steps. Applying one single γ can create a bias in quantifying the value of trajectories. Moreover, expert demonstrations may demonstrate a hybrid view of future rewards, *i.e.,* a hybrid discounting of trajectories via different discount factors. To address the limitation of using one single discount factor, we propose a new utility function to quantify the trajectory values by averaging the discounted cumulative rewards with different γ. Note that the average discounted cumulative rewards can be extended to weighted discounted cumulative rewards. For simplicity of presentation, we only focus on average discounted cumulative rewards throughout this paper.

Definition 1. *For a set Γ that contains different possible discount factors γ, the stereo utility of one trajectory τ^θ is defined as*

$$U_\phi(\tau^\theta) = \sum_{\gamma \in \Gamma} \frac{G_\phi(\tau^\theta, \gamma)}{|\Gamma|}, \tag{2}$$

where $|\Gamma|$ is the cardinality of Γ, ϕ is the parameter of the reward function, $G_\phi(\tau^\theta, \gamma)$ represents for the discounted cumulative reward of trajectory τ^θ with the discount factor γ, and θ is the parameter of the action policy.

The stereo utility can be interpreted as an expectation of the discounted cumulative reward with respect to the discount factor set Γ (with a uniform distribution). If an appropriate Γ is chosen, $U_\phi(\tau^\theta)$ is expected to provide a multi-spectrum view and interpretation of the demonstration, which is not achievable via the standard discounted cumulative reward $G_\phi(\tau^\theta, \gamma)$ when the true fixed discount factor is unknown.

One of the fundamental methods for IRL is maximum margin optimization which aims at learning a reward function that makes the demonstrations quantitatively better than alternative policies by a margin (Arora & Doshi, 2018). To enable the learning agent to exceed the demonstrator, we propose a new structure that is different from the standard maximum margin formulations where the demonstrator is assumed to always outperform the learning agent. In particular, we minimize the stereo utility difference between the learning agent's current trajectory and the demonstration. The minimum margin formulation with the same initial state s_0 is denoted as $\min_\phi \left(U_\phi(\tau_{s_0}^{\theta^+}) - U_\phi(\tau_{s_0}^{\theta^*})\right)$.

The above formulation is equivalent to maximum the gap between the expert's demonstration and the learning agent's trajectory. However, the philosophy here is to push the learning agent to catch up with the demonstrator. In order to let the learning agent achieve better performance while avoiding the divergence, we further propose to revise it as

$$\min_\phi \left[\lambda\left(U_\phi(\tau_{s_0}^{\theta^+}) - U_\phi(\tau_{s_0}^{\theta^*})\right) - G_\phi(\tau^\theta, \gamma)\right], \tag{3}$$

where $0 \leq \lambda < 1$ is the weight that determines the importance of minimizing the margin $U_\phi(\tau_{s_0}^{\theta^+}) - U_\phi(\tau_{s_0}^{\theta^*})$, and $G_\phi(\tau^\theta, \gamma)$ is the discounted cumulative reward of trajectory τ^θ calculated with a predefined γ in the policy optimization process. Note that $G_\phi(\tau^\theta, \gamma)$ can also be replaced by the stereo utility given in (2), which will be similar to a multi-horizons RL (Fedus et al., 2019).

Note that the trajectories $\tau_{s_0}^{\theta^+}$ and τ^θ are sampled from the learning agent's different policies. Specifically, $\tau_{s_0}^{\theta^+}$ is sampled from the most updated policy while τ^θ can be sampled from the past policies during the policy optimization process. The benefit of sampling the trajectories from different polices in (3) is that we can integrate the reward function learning process with the policy search process. In particular, we integrate the policy gradient method presented in (Sutton & Barto, 2018) with the optimization problem in (3) and propose a new loss function to solve the RL-MDP$\backslash R$ problem as

$$L(\theta, \phi) = -(1-\rho)\mathbb{E}\left[G_\phi(\tau^\theta, \gamma); \pi_\theta\right] + \rho\left[U_\phi(\tau_{s_0}^{\theta^+}) - U_\phi(\tau_{s_0}^{\theta^*})\right], \tag{4}$$

where $\rho \in [0, 1)$ is equivalent to the weight defined in (3) as $\lambda = \frac{\rho}{1-\rho}$.

Based on the loss function presented in (4), this subsection will present an algorithm that concurrently updates the reward function parameter ϕ and the action policy parameter θ. In particular, we propose a reward and policy co-learning (RPCL) algorithm that encloses the reward function learning process in the loop of policy optimization by updating ϕ occasionally with respective to θ.

Following the philosophy that a novice needs more guidance and an expert requires less instructions, we update ϕ more frequently during the early learning stage and gradually reduce the frequency. In particular, we select the Fibonacci sequence as the tool to change the ϕ updating frequency by leveraging the growing gap between two adjacent elements in the sequence. Let the ith updated ϕ be denoted as ϕ_i (ϕ_0 means the initial ϕ). ϕ remains unchanged as ϕ_i until the $(i+1)$th update. Hence, ϕ is independent of θ. Note that θ^+ is the most (recently) updated version of θ, which is constant and hence also independent of the θ learning process. Therefore, the partial partial derivative of $L(\theta, \phi)$ with respect to θ is only related to the first term in (4), *i.e.*, $\frac{\partial L(\theta,\phi)}{\partial \theta} = -\frac{\partial \mathbb{E}\left[G_\phi(\tau^\theta,\gamma);\pi_\theta\right]}{\partial \theta}$. The weight $1-\rho$ is ignored as the second weighted term is not included here and the learning rate itself can incorporate the extra weight parameter. Moreover, the gradient formulation will be the same

as gradient ascent based on the discounted cumulative reward (Sutton et al., 2000). Hence, we can choose an advantage actor-critic (Mnih et al., 2016) method to calculate $\frac{\partial L(\theta,\phi)}{\partial \theta}$. Specifically, the gradient can be calculated as

$$\frac{\partial L(\theta,\phi)}{\partial \theta} = -\sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t|s_t)\hat{A}_t, \tag{5}$$

where $\hat{A}_t$ denotes the advantage function and is calculated by $\hat{A}_t = \sum_{j=t}^{T} \gamma^{j-t} g(s_j|\phi) - \hat{V}(s_t)$, and $\hat{V}(s_t)$ is the estimated value function (Sutton & Barto, 2018) generated by the critic network. After θ has been updated for a number of times (which is determined by the ϕ updating frequency), the current θ is assigned to θ^+, which is then followed by the ϕ update process. Before explaining the detailed technique for updating ϕ, it is worth mentioning the benefit of the co-learning structure. As the reward function is updated less frequently than the policy parameter, the percentage of required demonstrations is small. In particular, let the maximum learning episode of the policy searching process be denoted as E. The number of demonstrations requires by RPCL is at most the total element number in the Fibonacci sequence, which is less than E.

During the above policy learning process, the reward function is constant. After a number of iterations, we update ϕ for K times based on the partial derivative of $L(\theta,\phi)$ with respect to ϕ as

$$\frac{\partial L(\theta,\phi)}{\partial \phi} = -(1-\rho)\frac{\partial \mathbb{E}\left[G_\phi(\tau^\theta,\gamma);\pi_\theta\right]}{\partial \phi} + \frac{\rho}{K}\frac{\partial D_i}{\partial \phi}, \tag{6}$$

where $D_i = U_\phi(\tau_{s_0}^{\theta^+}) - U_\phi(\tau_{s_0}^{\theta^*})$, and the expectation of the discounted cumulative reward can be approximated via samples as $\mathbb{E}\left[G_\phi(\tau^\theta,\gamma);\pi_\theta\right] = \sum_{\tau^\theta} p(\tau^\theta) G_\phi(\tau^\theta,\gamma)$, where $p(\tau^\theta) = p(\tau^\theta|\theta)p(\theta)$ denotes the possibility of obtaining the trajectory of τ^θ. Since the gradient $\frac{\partial L(\theta,\phi)}{\partial \theta}$ uses an online learning technique, the probability of having τ^θ for a given θ is one, *i.e.*, $p(\tau^\theta|\theta) = 1$, and $p(\theta)$ can be simplified as a uniform function. Let the number of sampled trajectories from the policy optimization process be n. Then we have $\sum_{\tau^\theta} p(\tau^\theta) G_\phi(\tau^\theta,\gamma) \approx \frac{1}{n}\sum_{k=1}^{n} G_\phi(\tau^{\theta_k},\gamma)$, where τ^{θ_k} represents the kth sampled trajectory from a policy π_{θ_k}. Hence, the gradient of ϕ in (6) can be approximated as

$$\frac{\partial L(\theta,\phi)}{\partial \phi} = -\frac{1-\rho}{n}\sum_{k=1}^{n}\sum_{j=1}^{T} \gamma^{j-1} \nabla_\phi g(s_j^k|\phi) + \frac{\rho}{K}\sum_{i=1}^{K} \nabla_\phi D_i, \tag{7}$$

where s_j^k is the state s_j in the kth sampled trajectory and n is the number of total sampled trajectories. The second term's derivative is given by $\nabla_\phi D_i = \frac{1}{|\Gamma|}\sum_{\gamma\in\Gamma}\left(\sum_{j=1}^{T}\gamma^{j-1}\nabla_\phi g(s_j^+|\phi) - \sum_{j=1}^{T}\gamma^{j-1}\nabla_\phi g(s_j^*|\phi)\right)$ for $s_j^+ \in \tau_{s_0}^{\theta^+}$ and $s_j^* \in \tau_{s_0}^{\theta^*}$.

4 Algorithm Evaluation

To evaluate the effectiveness of the proposed RPCL algorithm, we conduct experiments in three OpenAI gym environments (Brockman et al., 2016) which are *Cart Pole* (where the objective is to to balance a pole vertically by applying appropriate forces to the cart), *Mountain Car* (which trains an agent to navigate a car to the top of a hill by applying appropriate actions), and *Bipedal Walker* (where the goal is to train an agent to control a simulated bipedal walker to navigate through a terrain while maintaining balance and achieving a goal). We also conduct an indoor drone test to evaluate our approach in real world scenario. The indoor drone test is conducted with OptiTrack system which can provide real-time postions of the drone.

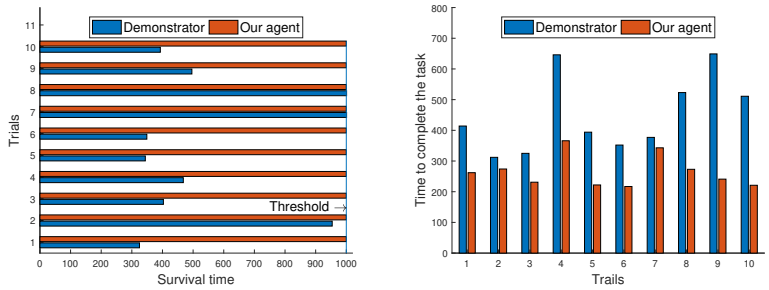
There are two available actions in *Cart Pole*, which are pushing the cart to the left ($a = 0$) or right ($a = 1$). The performance of a policy is proportional to the episode length, *i.e.*, the longer run time is, the better control policy is. *Mountain Car* has two versions, *i.e.*, *MountainCar-v0* and *MountainCarContinuous-v0*, which supports discrete and continuous actions respectively. In

particular, *MountainCar-v0* only has three discrete action inputs, namely, push left ($a = 0$), no push ($a = 1$), and push right ($a = 2$), while *MountainCarContinuous-v0* allows the input action to be continuous. For both versions, the episode termination condition is selected as the case when the cart reaches the top of the rightmost hill. If the condition is not satisfied, the episode will keep on running unless other extra stop conditions are met (*e.g.*, an upper limit of the experiment time). Different from *Cart Pole* which considers a policy's performance proportional to the run time, *Mountain Car* considers performance inverse proportional to the run time. In other words, the less steps it takes for the car to reach the goal position, the better the policy is. We discard the reward values from all these environments as they are assumed to be unknown in the RL-MDP$\backslash R$ setting. To obtain expert policies, we adopt LQR method on the linearized model of *Cart Pole* as a demonstrator. We adopt the policies learned from the policy search with the original reward for two *Mountain Car* environments as the expert demonstrators since there are no analytical solutions. With the expert demonstrators, we then implement the proposed RPCL algorithm in all these environments, where the policy and reward functions are approximated by neural networks.

To verify the performance of the proposed RPCL algorithm in these environments, we evaluate the learned policies against the demonstrators' policies and the policies (using the sample actor-critic RL structure as ours) learned from environment reward. We let all the three polices share the same initial state and run experiments for 1000 random initial states. As can be observed in Table 1, our RPCL agent outperforms the other two policies in all these environments. A random selected 10 trials of performances for both our RPCL agents and expert demonstrators in *Cart Pole* and *Mountain Car* are shown in Fig. 1.

Table 1: Results comparison for *CartPole-v0* (CP), *MountainCar-v0* (MC), and *MountainCarContinuous-v0* (MCC) with respect to the running steps

Environment	RPCL	Expert	AC with Env Reward
CP	**1000±0**	716±310	972 ±148
MC	**135±22**	253±143	135±32
MCC	**273±61**	417±127	393±120



(a) CartPole (b) Mountain Car

Figure 1: Performance comparison between the expert demonstrator and the learned policy from the RPCL algorithm in the *CartPole* and *Mountain Car* environments.

We explore the choice of different discount factor sets and the corresponding effect on the agent's performance. The results show that the inclusion of more discount factors in the form of the proposed stereo utility can generally yield improved performance. We also test our RPCL with *Bipedal Walker* environment which has a higher dimensional task to be solved. In particular, the *BipedalWalker-v3* environment simulates the bipedal locomotion, which takes 4 degrees of freedom (2 hip and knee joints) action inputs and provides 24-dimensional state information which consists of kinematic parameters of the agent and 10 lidar rangefinder measurements. The performance of the agent is not directly related to the run time. Instead, the performance is measured by whether the agent moves to the far end without falling and how much motor torque needs to be applied.

In our RPCL simulation, we adopt a policy learned from the policy search with the original reward, obtained via DDPG (Lillicrap et al., 2015), as the expert demonstrators. The demonstrator can achieve 288 ± 69 evaluated over 500 rollouts. With this expert demonstrator, we then implement PRCL in *Bipedal Walker*. For a 500-rollout statistical evaluation, our RPCL policy can get a score of 298±27, which outperforms the demonstrator.

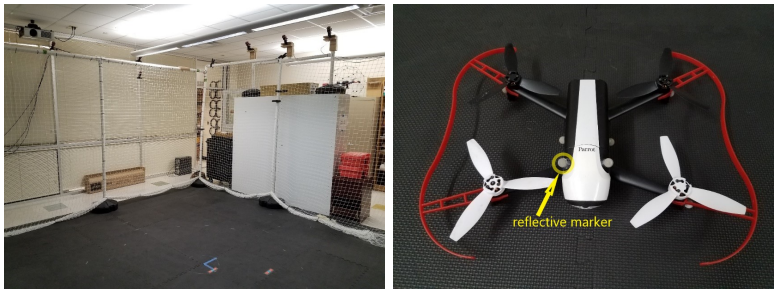
To further show the advantages of the proposed RPCL method, we also perform comparison of the proposed RPCL method with other baseline methods, including behavior cloning, generative adversarial imitation learning (GAIL), maximum margin inverse reinforcement learning, and maximum-entropy inverse reinforcement learning. Table 2 shows the outcomes of the conducted comparisons. It can be seen that our RPCL algorithm outperforms Demonstrator, Behavior cloning, Maximum Margin, RL with environment reward, and GAIL in all examples. We were only able to obtain one example (MountainCar-discrete) that maximum-entropy IRL applies. The maximum-entropy IRL works better than ours, while requiring 5 times more episodes of training.

Table 2: Comparison with other baseline methods using the environment reward

IRL methods	CP	MC	MCC	BW
Behaviour Cloning	886±183	Fails	72±41	287±75
RPCL	**1000±0**	-143±36	**84±6**	**298±27**
RL with Env reward	972±148	-162±41	82±13	290±25
GAIL	36±22*	N/A	N/A	255±123
Maximum Margin	983±122	-153±34	81±15	274±1
MaxEnt	N/A	**-123± 1**[1]	N/A	N/A
Demonstrator	798±287	-247±135	76±8	288±69

* obtained in the wrapped environment setting (maximum 200 steps).
[1] requires 30000 episodes of training.

To verify the performance of our RPCL algorithm in real world, we test it under a physical indoor drone testing environment, where the environment rewards are unavailable. Fig. 2 shows the indoor Bebop drone testing environment with OptiTrack system (Wu et al., 2023). The OptiTrack system can provide position and velocity information of the drone. Since the OptiTrack system can only track objects through 18 infrared cameras, the Bebop drone was modified via adding 6 reflective markers that can be captured by the 18 infrared cameras. The 6 reflective markers are attached to the drone at different locations to form a rigid body that can be further used as the identification of the drone in the Motive, which is an optical motion capture software. In particular, we conduct a simple task by driving the drone from a given ground position (-1.2±0.1m, 0.2±0.1m, 0) to the origin of 3D coordinate system, *i.e.*, (0, 0, 0). The control input for the drone is discretized into seven possible actions, *i.e.*, stay still, move forward/backward, move left/right, and rise/fall.



(a) OptriTrack system (b) Modified Drone

Figure 2: Indoor Bebop drone testing setup.

We choose a PI controller as the demonstrator. More precisely, the PI controller will first navigate the drone parallel to the ground when the drone takes off. The movement in z axis is only activated

when the drone is above the acceptable landing area. In our tests, we set the acceptable landing area as a circle with a radius of 0.25 meters around the origin. To train our RPCL algorithm, we select an actor-critic algorithm and take the x, y, z positions and v_x, v_y, v_z velocities as the input states. The parameters of our RPCL are set as $\rho = 0.99,\ \eta = 0.99,\ \gamma = 0.99, \Gamma = [0.9, 0.995],\ \epsilon_1 = 5e-5,\ \epsilon_2 = 0.01,\ n = 1$, and $N = 1000$. The variables e and E are not set because operating large real flight tests is time-consuming. Hence, the stop condition is determined by the operator. In our experiments, we conduct 279 episodes of flight and only 14 demonstrations are required.

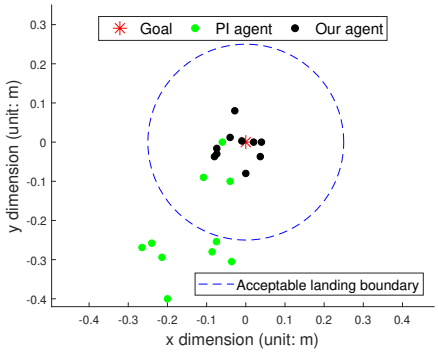


Figure 3: Performance comparison between the demonstrator and the RPCL algorithm for indoor real drone testing.

Table 3: Landing Statistic Comparison (unit: meter)

Controller	PI agent	RPCL agent
Mean	0.27	0.057
Standard deviation	0.13	0.028

As we observe, the drone already performs well when training episode reaches 270. Hence, we stop the training at episode 279 and then compare the derived policy with the PI controller. 10 flights are conducted and the landing markers for our agent and the PI controller are plotted in Fig. 3. It can be seen that our agent performs better and more stable than the PI controller does. The statistics of the landing distance with respect to the goal are shown in Table 3. In particular, the distance error for our RPCL agent is $0.057 \pm 0.028m$ while the PI agent has a high error of $0.27 \pm 0.13m$. In other words, the proposed RPCL algorithm can reduce the landing error by approximately 80% by learning from and then exceeding the PI controller.

5 Conclusion and Future Work

In this paper, we proposed a new reward and control policy co-learning (RPCL) algorithm to derive control policies that can mimic and outperform expert's demonstrations in Markov decision processes, where the reward function is unknown. The RPCL algorithm is built on the construction of a new stereo utility function and the design of a new loss function. To evaluate the performance of RPCL, we conduct experimental studies in three virtual environments and a physical indoor drone flight environment.

One interesting future research direction is to investigate the application of RPCL in more complex environments when exceeding expert demonstrations can be more challenging due to, *e.g.*, environment nonconvexity or the existence of multiple objectives.

Acknowledgements

The work was supported in part by Army Research Office under grants W911NF2110103 and W911NF2310363, and Office of Naval Research under grant N000142212474.

References

Saurabh Arora and Prashant Doshi. A survey of inverse reinforcement learning: Challenges, methods and progress. *arXiv:1806.06877*, 2018.

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *arXiv:1606.01540*, 2016.

Daniel S Brown, Wonjoon Goo, Prabhat Nagarajan, and Scott Niekum. Extrapolating beyond suboptimal demonstrations via inverse reinforcement learning from observations. *arXiv preprint arXiv:1904.06387*, 2019.

Daniel S Brown, Wonjoon Goo, and Scott Niekum. Better-than-demonstrator imitation learning via automatically-ranked demonstrations. In *Conference on Robot Learning*, pp. 330–359, 2020.

JD Choi and Kee-Eung Kim. Inverse reinforcement learning in partially observable environments. *Journal of Machine Learning Research*, 12:691–730, 2011.

Rati Devidze, Parameswaran Kamalaruban, and Adish Singla. Exploration-guided reward shaping for reinforcement learning under sparse rewards. *Advances in Neural Information Processing Systems*, 35:5829–5842, 2022.

Tom Everitt, Victoria Krakovna, Laurent Orseau, Marcus Hutter, and Shane Legg. Reinforcement learning with a corrupted reward channel. *arXiv:1705.08417*, 2017.

William Fedus, Carles Gelada, Yoshua Bengio, Marc G Bellemare, and Hugo Larochelle. Hyperbolic discounting and learning over multiple horizons. *arXiv preprint arXiv:1902.06865*, 2019.

Chelsea Finn, Sergey Levine, and Pieter Abbeel. Guided cost learning: Deep inverse optimal control via policy optimization. In *Proceedings of the International Conference on Machine Learning*, pp. 49–58, 2016.

Yujing Hu, Weixun Wang, Hangtian Jia, Yixiang Wang, Yingfeng Chen, Jianye Hao, Feng Wu, and Changjie Fan. Learning to utilize shaping rewards: A new approach of reward shaping. *Advances in Neural Information Processing Systems*, 33:15931–15941, 2020.

Yunhan Huang and Quanyan Zhu. Deceptive reinforcement learning under adversarial manipulations on cost signals. *arXiv:1906.10571*, 2019.

Yuqian Jiang, Suda Bharadwaj, Bo Wu, Rishi Shah, Ufuk Topcu, and Peter Stone. Temporal-logic-based reward shaping for continuing reinforcement learning tasks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 7995–8003, 2021.

Bingyi Kang, Zequn Jie, and Jiashi Feng. Policy optimization with demonstrations. In *International conference on machine learning*, pp. 2469–2478. PMLR, 2018.

Arun Kumar, Navneet Paul, and SN Omkar. Bipedal walking robot using deep deterministic policy gradient. *arXiv:1807.05924*, 2018.

Sergey Levine and Pieter Abbeel. Learning neural network policies with guided policy search under unknown dynamics. In *Advances in Neural Information Processing Systems*, pp. 1071–1079. 2014.

Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529, 2015.

Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *Proceedings of the International Conference on Machine Learning*, pp. 1928–1937, 2016.

Ashvin Nair, Bob McGrew, Marcin Andrychowicz, Wojciech Zaremba, and Pieter Abbeel. Overcoming exploration in reinforcement learning with demonstrations. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 6292–6299, 2018.

Andrew Y Ng, Daishi Harada, and Stuart Russell. Policy invariance under reward transformations: Theory and application to reward shaping. In *Proceedings of the International Conference on Machine Learning*, volume 99, pp. 278–287, 1999.

Andrew Y Ng, Stuart J Russell, et al. Algorithms for inverse reinforcement learning. In *Proceedings of the International Conference on Machine Learning*, volume 1, pp. 2, 2000.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv:1707.06347*, 2017.

Kyriacos Shiarlis, Joao Messias, and Shimon Whiteson. Inverse reinforcement learning from failure. In *Proceedings of the 2016 International Conference on Autonomous Agents & Multiagent Systems*, pp. 1060–1068, 2016.

David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484, 2016.

Adrian Šošić, Abdelhak M Zoubir, Elmar Rueckert, Jan Peters, and Heinz Koeppl. Inverse reinforcement learning via nonparametric spatio-temporal subgoal modeling. *Journal of Machine Learning Research*, 19(1):2777–2821, 2018.

Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

Richard S Sutton, David A McAllester, Satinder P Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Advances in Neural Information Processing Systems*, pp. 1057–1063, 2000.

Devin White, Mingkang Wu, Ellen Novoseller, Vernon J Lawhern, Nicholas Waytowich, and Yongcan Cao. Rating-based reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 10207–10215, 2024.

Mingkang Wu, Feng Tao, and Yongcan Cao. Value of potential field in reward specification for robotic control via deep reinforcement learning. In *AIAA SCITECH 2023 Forum*, pp. 0505, 2023.

Zheng Wu, Liting Sun, Wei Zhan, Chenyu Yang, and Masayoshi Tomizuka. Efficient sampling-based maximum entropy inverse reinforcement learning with application to autonomous driving. *IEEE Robotics and Automation Letters*, 5(4):5355–5362, 2020.

Markus Wulfmeier, Peter Ondruska, and Ingmar Posner. Maximum entropy deep inverse reinforcement learning. *arXiv:1507.04888*, 2015.

Brian D Ziebart, Andrew Maas, J Andrew Bagnell, and Anind K Dey. Maximum entropy inverse reinforcement learning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 2008.