Receding Horizon Inverse Reinforcement Learning

Yiqing Xu[1] **Wei Gao**[1] **David Hsu**[1,2]

[1]School of Computing
[2]Smart Systems Insitute
National University of Singapore
{xuyiqing,gaowei90,dyhsu}@comp.nus.edu.sg

Abstract

Inverse reinforcement learning (IRL) seeks to infer a cost function that explains the underlying goals and preferences of expert demonstrations. This paper presents *receding-horizon inverse reinforcement learning* (RHIRL), an IRL algorithm for high-dimensional, noisy, continuous systems with black-box dynamic models. RHIRL addresses two key challenges of IRL: scalability and robustness. To handle high-dimensional continuous systems, RHIRL matches the induced optimal trajectories with expert demonstrations *locally* in a receding horizon manner and "stitches" together the local solutions to learn the cost; it thereby avoids the "curse of dimensionality". This contrasts with earlier algorithms that match with expert demonstrations *globally* over the entire high-dimensional state space. To be robust against imperfect expert demonstrations and control noise, RHIRL learns a state-dependent cost function "disentangled" from system dynamics under mild conditions. Experiments on benchmark tasks show that RHIRL outperforms several leading IRL algorithms in most instances. We also prove that the cumulative error of RHIRL grows linearly with the task duration.

1 Introduction

Reinforcement learning (RL) has made exciting progress in a range of complex tasks, including real-time game playing [24], visuo-motor control of robots [35], and many other works. The success, however, often hinges on a carefully crafted cost function [17, 25], which is a major impediment to the wide adoption of RL in practice. Inverse reinforcement learning (IRL) [26] addresses this need by learning a cost function that explains the underlying goals and preferences of expert demonstrations. This work focuses on two key challenges in IRL, *scalability* and *robustness*.

Classic IRL algorithms commonly consist of two nested loops. The inner loop approximates the optimal control policy for a hypothesized cost function, while the outer loop updates the cost function by comparing the behavior of the induced policy with expert demonstrations. The inner loop must solve the (forward) reinforcement learning or optimal control problem, which is in itself a challenge for complex high-dimensional systems. Many interesting ideas have been proposed for IRL, including, e.g., maximum entropy learning [39, 41], guided cost learning [7], and adversarial learning [8]. See Figure 1 for illustrations. They try to match a *globally* optimal approximate policy with expert demonstrations over the entire system state space or a sampled approximation of it. This is impractical for high-dimensional continuous systems and is a fundamental impediment to scalability. Like RL, IRL suffers from the same "curse of dimensionality". To scale up, *receding-horizon IRL* (RHIRL) computes locally optimal policies with receding horizons rather than a globally optimal policy and then matches them with expert demonstrations *locally* in succession (Figure 1*d*). The local approximation and matching substantially mitigate the impact of high-dimensional space and improve the sample efficiency of RHIRL, at the cost of a local rather than a global solution. So RHIRL trades off optimality for scalability and provides an alternative to current approaches.

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

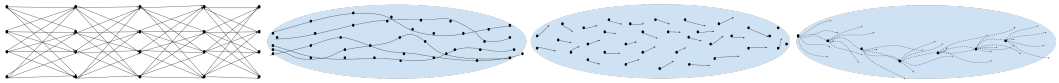


Figure 1: A comparison of RHIRL and selected IRL algorithms. They all try to match the policy induced by the learned cost with expert demonstrations. (*a*) MaxEnt matches the exact feature count over the entire system state space. (*b*) REIRL, GCL, and GAN-GCL match the approximate feature count over sampled state trajectories globally over the entire task duration. (*c*) GAIL, AIRL, ... discriminate the sampled state or state-action distributions. (*d*) RHIRL matches control sequences *locally* along demonstrated trajectories in a receding horizon manner.

Another important concern of IRL is noise in expert demonstrations and system control. Human experts may be imperfect for various reasons and provide good, but still suboptimal demonstrations. Further, the system may fail to execute the commanded actions accurately because of control noise. We want to learn a cost function that captures the expert's intended actions rather than the imperfectly executed actions. While earlier work has investigated the question of learning from sub-optimal or failed demonstrations [3, 31, 38], there is a subtle, but critical difference between (i) demonstrations intended to be suboptimal and (ii) optimal demonstrations corrupted by noise. The existing work [3, 31, 38] addresses (i); RHIRL addresses (ii). To learn the true intentions from the noise corrupted demonstrations, RHIRL relies on a simplifying assumption: the cost function is linearly separable with two components, one state-dependent and one control-dependent. Many interesting systems in practice satisfy the assumption, at least, approximately [28]. RHIRL then learns the state-dependent cost component, which is disentangled from the system dynamics [8] and agnostic to noise.

2 Related Work

IRL can be viewed as an indirect approach to imitation learning. It learns a cost function, which induces an optimal policy whose behavior matches with expert demonstrations. In contrast, behavior cloning (BC) is a direct approach. It assumes independence among all demonstrated state-action pairs and learns a policy that maps states to actions through supervised learning of state-action pairs from expert demonstrations. The simplicity of BC is appealing. However, it typically requires large amounts of data to learn well and suffers from covariate shift [30]. IRL is more data-efficient. Further, it produces a cost function, which explains the expert demonstrations and potentially transfers to other systems with different dynamics. These benefits, however, come at the expense of greater computational complexity.

Classic IRL algorithms learn a cost function iteratively in a double-loop: the outer loop updates a hypothesized cost function, and the inner loop solves the forward RL problem for an optimal policy and matches it with expert demonstrations. Various methods have been proposed [1, 7, 39, 41], but they all seek a globally optimal solution over the entire state space (Figure 1*a*) [39, 41] or the entire task duration (Figure 1*b*) [1, 7]. As a result, they face significant challenges in scalability and must make simplifications, such as locally linear dynamics [7]. Recent methods use the generative adversarial network (GAN) [10] to learn a discriminator that differentiates between the state or state-action distribution induced by the learned cost and that from expert demonstrations [5, 8, 9, 13, 14, 18, 21, 27, 29]. We view this global matching as a major obstacle to scalability. In addition, GAN training is challenging and faces difficulties with convergence.

RHIRL stands in between BC and the aforementioned IRL algorithms by trading off optimality for scalability. BC performs local matching at each demonstrated state-action pair, treating all of them independently. Most existing IRL algorithms perform global matching over the entire state space or sampled trajectories from it. RHIRL follows the standard IRL setup. To tackle the challenge of scalability for high-dimensional continuous systems, RHIRL borrows ideas from *receding horizon control* (RHC) [19]. It solves for locally optimal control sequences with receding horizons and learns the cost function by "stitching" together a series of locally optimal solutions to match the global state distribution of expert demonstrations (Figure 1*d*). Earlier work has explored the idea of RHC in IRL [23], but relies on handcrafted reward features and is limited to discrete, low-dimensional tasks. More recent work learns reward features automatically [20]. However, it focuses on lane navigation for autonomous driving, exploting a known analytic model of dynamics and noise-free,

perfect expert demonstrations. RHIRL aims at general high-dimensional continuous tasks with noisey expert demonstrations.

Another challenge to IRL is suboptimal expert demonstrations and system control noise. Several methods learn an auxiliary score or ranking to reweigh the demonstrations, in order to approximate the underlying optimal expert distribution [4, 5, 38]. RHIRL does not attempt to reconstruct the optimal expert demonstrations. It explicitly models the human errors in control actions as additive Gaussian noise and matches the noisy control with expert demonstrations, in order to learn from the intended, rather than the executed expert actions. Modelling the human error as additive Gaussian noise is a natural choice technically [22, 34, 40], but compromises on realism. Human errors in sequential decision making may admit other structural regularities, as a result of planning, resource constraints, uncertainty, or bounded rationality[15]. Studying the specific forms of human errors in the IRL context requires insights beyond the scope of our current work, but forms a promising direction for future investigation.

3 Receding Horizon Inverse Reinforcement Learning

3.1 Overview

Consider a continuous dynamical system:

$$x_{t+1} = f(x_t, v_t), \tag{1}$$

where $x_t \in \mathcal{R}^n$ is the state, $v_t \in \mathcal{R}^m$ is the control at time t, and the initial system state at $t = 0$ follows a distribution μ. To account for noise in expert demonstrations, we assume that v_t is a random variable following the Gaussian distribution $\mathcal{N}(v_t|u_t, \Sigma)$, with mean u_t and covariance Σ. We can control u_t directly, but not v_t, because of noise. The state-transition function f captures the system dynamics. RHIRL represents f as a black-box simulator and does not require its analytic form. Thus, we can accommodate arbitrary complex nonlinear dynamics. To simplify the presentation, we assume that the system dynamics is deterministic. We sketch the extension to stochastic dynamics at the end of the section, the full proof is given in Appendix C.

In RL, we are given a cost function and want to find a control policy that minimizes the expected total cost over time under the dynamical system. In IRL, we are not given the cost function, but instead, a set $\mathcal{D}$ of expert demonstrations. Each demonstration is a trajectory of states visited by the expert over time: $\tau = (x_0, x_1, x_2, ..., x_{T-1})$ for a duration of T steps.

We assume that the expert chooses the actions to minimize an unknown cost function and want to recover this cost from the demonstrations. Formally, suppose that the cost function is parameterized by θ. RHIRL aims to learn a cost function that minimizes the loss $\mathcal{L}(\theta; \mathcal{D})$, which measures the difference between the demonstration trajectories and the optimal control policy induced by the cost function with parameters θ.

RHIRL performs this minimization iteratively, using the gradient $\partial\mathcal{L}/\partial\theta$ to update θ. In iteration t, let x_t be the system state at time t and $\mathcal{D}_t$ be the set of expert sub-trajectories starting at time t and having a duration of maximum K steps. We use the current cost to perform receding horizon control (RHC) at x_t, with time horizon K, and then update the cost by comparing the resulting state trajectory distribution with the demonstrations in $\mathcal{D}_t$. Earlier IRL work requires the simulator to reset to the states from the initial state distribution [1, 6, 7, 8, 13, 18]. To perform receding horizon optimization, RHIRL requires a simulator to reset to the initial state x_t for the local control sequence optimization at each time step $t = 0, \ldots T-1$, a common setting in optimal control (e.g., [36, 37, 20]). See Algorithm 1 for a sketch.

First, sample M control sequences, each of length K (line 6). We assume that the covariance Σ is known. If it is unknown, we set it to be identity by default. For our experiment results reported in Table 1 and Table 2, Σ is unknown and is approximated by a constant factor of the identity matrix (grid search is performed to determine this constant factor). We show through experiments that the learned cost function is robust over different noise settings (Section 4.3). Next, we apply model-predictive path integral (MPPI) control [37] at x_t. MPPI provides an analytic solution for the optimal control sequence distribution and the associated state sequence distribution, which allow us to estimate the gradient $\partial\mathcal{L}/\partial\theta$ efficiently through importance sampling (lines 7–8) and update cost function parameters θ (line 9). Finally, we execute the computed optimal control (line 10–12)

Algorithm 1 RHIRL

1: Initialize θ randomly.
2: **for** $i = 1, 2, 3, ...$ **do**
3: Sample x_0 from μ.
4: Initialize control sequence U of length K to $(0, 0, \dots)$.
5: **for** $t = 0, 1, 2, ..., T-1$ **do**
6: Sample M control sequences V_j for $j = 1, 2, \dots M$, according to $\mathcal{N}(V|U, \Sigma)$.
7: Compute the importance weight w_j for each V_j, using the state cost $S(V_j, x_t; \theta)$. See equation (11).
8: Compute $\frac{\partial}{\partial\theta}\mathcal{L}(\theta; \mathcal{D}_t, x_t)$ using $\mathcal{D}_t$ and V_j with weight w_j, for $j = 1, 2, \dots, M$. See equation (12).
9: $\theta \leftarrow \theta - \alpha \frac{\partial \mathcal{L}}{\partial \theta}$.
10: $U \leftarrow \sum_{j=1}^{M} w_j V_j$.
11: Sample v_t from $\mathcal{N}(v|u, \Sigma)$, where u is the first element in the control sequence U.
12: $x_{t+1} \leftarrow f(x_t, v_t)$.
13: Remove u from U. Append 0 at the end of U.
14: **end for**
15: **end for**

and update the mean control input for the next iteration (line 13). We would like to emphasize that we only use the simulator to sample rollouts and evaluate the current cost function. Fixed expert demonstrations $\mathcal{D}$ are given upfront. Unlike DAgger, we do not query the expert online during training.

Our challenge is to uncover a cost function that captures the expert's intended controls $(u_0, u_1, \dots, u_{T-1})$, even though they were not directly observed, because of noise, and do so in a scalable and robust manner for high-dimensional, noisy systems.

We develop three ideas: structuring the cost function, matching locally with expert demonstrations, and efficient computation of the gradient $\partial\mathcal{L}/\partial\theta$, which are described next.

3.2 Robust Cost Functions

To learn a cost function robust against noise, we make a simplifying assumption that linearly separates the one-step cost into two components: a state cost $g(x; \theta)$ parameterized by θ and a quadratic control cost $u^T\Sigma^{-1}u$. Despite the simplification, this cost function models a wide variety of interesting systems in practice [28]. It allows RHIRL to learn a state cost $g(x; \theta)$, independent of control noise (Section 3.4), and thus generalize over different noise distributions (Section 4.3).

Suppose that $V = (v_0, v_1, \dots, v_{K-1})$ is a control sequence of length K, conditioned on the input $U = (u_0, u_1, \dots, u_{K-1})$. We apply V to the dynamical system in (1) with start state x_0 and obtain a state sequence $\tau = (x_0, x_1, \dots, x_K)$ with $x_k = f(x_{k-1}, v_{k-1})$ for $k = 1, 2, \dots, K$. Define the total cost of V as

$$J(V, x_0; \theta) = \sum_{k=0}^{K} g(x_k; \theta) + \sum_{k=0}^{K-1} \frac{\lambda}{2} u_k^\intercal \Sigma^{-1} u_k, \tag{2}$$

where $\lambda \geq 0$ is a constant weighting the relative importance between the state and control costs. For convenience, define also the total state cost of V as

$$S(V, x_0; \theta) = \sum_{k=0}^{K} g(x_k; \theta). \tag{3}$$

While S is defined in terms of the control sequence V, it only depends on the corresponding state trajectory τ. This is very useful, as the training data contains state and not control sequences explicitly.

3.3 Local Control Sequence Matching

To minimize the loss $\mathcal{L}$, each iteration of RHIRL applies RHC with time horizon K under the current cost parameters θ and computes locally optimal control sequences of length K. In contrast, classic IRL algorithms, such as MaxEnt [41], perform global optimization over the entire task duration T in the inner loop. While RHC sacrifices global optimality, it is much more scalable and enables RHIRL

to handle high-dimensional continuous systems. We use the hyperparameter K to trade off optimality and scalability.

Specifically, we use MPPI [37] to solve for an optimal control sequence distribution at the current start state x_t in iteration t. The main result of MPPI suggests that the optimal control sequence distribution Q minimizes the "free energy" of the dynamical system and this free energy can be calculated from the cost of the state trajectory under Q. Mathematically, the probability density $p_Q(V^*)$ can be expressed as a function of the state cost $S(V, x_t; \theta)$, with respect to a Gaussian "base" distribution $B(U_B, \Sigma)$ that depends on the control cost:

$$p_Q(V^*|U_B, \Sigma, x_t; \theta) = \frac{1}{Z}\, p_B(V^*|U_B, \Sigma) \exp\big(-\frac{1}{\lambda} S(V^*, x_t; \theta)\big), \tag{4}$$

where Z is the partition function. For the quadratic control cost in (2), we have $U_B = (0, 0, \ldots)$ [37]. Intuitively, the expression in (4) says that the probability density of V^* is the product of two factors, one penalizing high control cost and one penalizing high state cost. So controls with large values or resulting in high-cost states occur with probability exponentially small.

Equation (4) provides the optimal control sequence distribution under the current cost. Suppose that the control sequences for expert demonstrations $\mathcal{D}_t$ follow a distribution E. We define the loss $\mathcal{L}(\theta; \mathcal{D}_t, x_t)$ as the KL-divergence between the two distributions:

$$\mathcal{L}(\theta; \mathcal{D}_t, x_t) = D_{\text{KL}}\big(p_E(V|x_t) \parallel p_Q(V|U_B, \Sigma, x_t; \theta)\big), \tag{5}$$

which RHIRL seeks to minimize in each iteration. While the loss $\mathcal{L}$ is defined in terms of control sequence distributions, the expert demonstrations $\mathcal{D}$ provide state information and not control information. However, each control sequence V induces a corresponding state sequence τ for a given start state x_0, and τ determines the cost of V according to (2). We show in the next subsection that $\partial\mathcal{L}/\partial\theta$ can be computed efficiently using only state information from $\mathcal{D}$. This makes RHIRL appealing for learning tasks in which control labels are difficult or impossible to acquire.

3.4 Gradient Optimization

To simplify notations, we remove the explicit dependency on x_t and $\mathcal{D}_t$ and assume that in iteration t of RHIRL, all control sequences are applied with x_t as the start state and expert demonstrations come from $\mathcal{D}_t$. We have

$$\begin{aligned}\frac{\partial\mathcal{L}}{\partial\theta} &= \frac{\partial}{\partial\theta}\int p_E(V)\log\frac{p_E(V)}{p_Q(V|U_B, \Sigma; \theta)}dV \\ &= \int p_E(V)\Big(\frac{1}{\lambda}\frac{\partial}{\partial\theta}S(V;\theta)\Big)dV - \int p_Q(V|U_B, \Sigma; \theta)\Big(\frac{1}{\lambda}\frac{\partial}{\partial\theta}S(V;\theta)\Big)dV\end{aligned} \tag{6}$$

The first line follows directly from the definition, and the derivation for the second line is available in Appendix A.

We estimate the two integrals in (6) through sampling. For the first integral, we can use the expert demonstrations as samples. For the second integral, we cannot sample p_Q directly, as the optimal control distribution Q is unknown in advance. Instead, we sample from a known Gaussian distribution with density $p(V|U, \Sigma)$ and apply importance sampling so that

$$\mathbb{E}_{p_Q(V|U_B,\Sigma;\theta)}[V] = \mathbb{E}_{p(V|U,\Sigma)}[w(V)V]. \tag{7}$$

The importance weight $w(V)$ is given by

$$w(V) = \frac{p_Q(V|U_B, \Sigma; \theta)}{p(V|U, \Sigma)} = \frac{p_Q(V|U_B, \Sigma; \theta)}{p_B(V|U_B, \Sigma)} \times \frac{p_B(V|U_B, \Sigma)}{p(V|U, \Sigma)} \tag{8}$$

To simplify the first ratio in (8), we substitute in the expression for p_Q from (4):

$$\frac{p_Q(V|U_B, \Sigma; \theta)}{p_B(V|U_B, \Sigma)} = \frac{1}{Z}\exp\Big(-\frac{1}{\lambda}(S(V;\theta)\Big) \tag{9}$$

We then simplify the second ratio, as both are Gaussian distributions with the same covariance matrix Σ:

$$\frac{p_B(V|U_B, \Sigma)}{p(V|U, \Sigma)} \propto \exp\Bigg(-\sum_{k=0}^{K-1}(u_k - u_k^B)^\intercal \Sigma^{-1} v_k)\Bigg), \tag{10}$$

where u_k and v_k, $k = 0, 1, \ldots, K-1$ are the mean controls and the sampled controls from $p(V|U, \Sigma)$, respectively. Similarly, u_k^B, $k = 0, 1, \ldots, K-1$ are the mean controls for the base distribution, and they are all 0 in our case. Substituting (9) and (10) into (8), we have

$$w(V) \propto \exp(-\frac{1}{\lambda}\left(S(V;\theta) + \lambda \sum_{k=0}^{K-1} u_k^\intercal \Sigma^{-1} v_k\right) \tag{11}$$

For each sampled control sequence V, we evaluate the expression in (11) and normalize over all samples to obtain $w(V)$.

To summarize, we estimate $\partial\mathcal{L}/\partial\theta$ through sampling:

$$\frac{\partial}{\partial\theta}\mathcal{L}(\theta; \mathcal{D}_t, x_t) \approx \frac{1}{N}\sum_{i=1}^{N}\frac{1}{\lambda}\frac{\partial}{\partial\theta}S(V_i, x_t; \theta) - \frac{1}{M}\sum_{j=1}^{M}\frac{1}{\lambda}w(V_j)\frac{\partial}{\partial\theta}S(V_j, x_t; \theta), \tag{12}$$

where $V_i, i = 1, \ldots, N$ are the control sequences for the expert demonstrations in $\mathcal{D}_t$ and $V_j, j = 1, 2, \ldots, M$ are the sampled control sequences. Equation (12) connects $\partial\mathcal{L}/\partial\theta$ with $\partial S/\partial\theta$. The state cost function S is represented as a shallow neural network, and its derivative can be obtained easily through back-propagation. To evaluate $\frac{\partial}{\partial\theta}S(V_i, x_t; \theta)$, we do not actually use the expert control sequences, as they are unknown. We use the corresponding state trajectories in $\mathcal{D}_t$ directly, as the state cost depends only on the visited states. See equation (3).

Finally, we approximate the optimal mean control sequence according to (7):

$$U = \mathbb{E}_{p_Q(V|U_B, \Sigma, x_t; \theta)}[V] \approx \sum_{j=1}^{M} w(V_j)V_j. \tag{13}$$

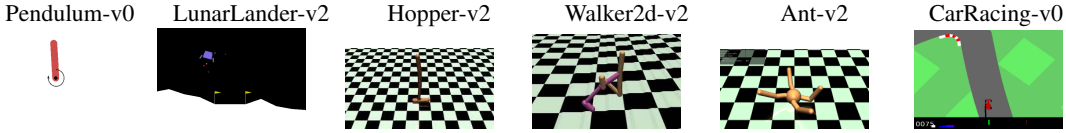
The first element in the control sequence U is the chosen control for the current time step t. We remove the first element from U and append zero at the end. The new control sequence is then used as the mean for the sampling distribution in the next iteration.

3.5 Analysis

Since RHIRL performs local optimization sequentially over many steps, one main concern is error accumulation over time. For example, standard behavior cloning has one-step error ϵ and cumulative error $O(T^2\epsilon)$ over T steps, because of covariate shift [30]. The DAgger algorithm reduces the error to $O(T\epsilon)$ by querying the expert repeatedly during online learning [30]. We prove a similar result for RHIRL, which uses offline expert demonstrations only. In iteration t of RHIRL, let $p_E(V_t|x_t)$ be the K-step expert demonstration distribution and $p_Q(V_t|U_B, \Sigma, x_t; \theta)$ be the computed K-step optimal control distribution for some fixed cost parameters θ. RHIRL minimizes the KL-divergence between these two distributions in each iteration. Let $p_{\text{E}}(x)$ be the state marginal distribution of expert demonstrations and $p_{\text{RHC}}(x; \theta)$ be the state marginal distribution of the computed RHC policy under θ over the entire task duration T. Intuitively, we want the control policy under the learned cost to visit states similar to those of expert demonstrations in distribution. In other words, $p_{\text{RHC}}(x; \theta)$ and $p_{\text{E}}(x)$ are close.

Theorem 3.1. *If* $D_{\text{KL}}\big(p_E(V_t|x_t) \parallel p_Q(V_t|U_B, \Sigma, x_t; \theta)\big) < \epsilon$ *for all* $t = 0, 1, ..., T-1$, *then* $D_{\text{TV}}(p_{\text{E}}(x), p_{\text{RHC}}(x; \theta)) < T\sqrt{\epsilon/2}$.

The theorem says that RHIRL's cumulative error, measured in total variation distance D_{TV} between $p_{\text{E}}(x)$ and $p_{\text{RHC}}(x; \theta)$, grows linearly with T. The proof consists of three steps. First, in each iteration t, if the KL-divergence between two control sequence distributions are bounded by ϵ, so is the KL-divergence between the two corresponding state distributions induced by control sequences. Next, we show that the KL-divergence between the state distributions over two successive time steps are bounded by the same ϵ. Finally, we switch from KL-divergence to total variation distance and apply the triangle inequality to obtain the final result. Note that our local optimization's objective is defined in KL divergence, while the final error bound is in TV distance. We switch the distance measures to get the best from both. Minimizing the KL divergence leads to strong local optimization result, but KL itself is not a proper metric. Therefore, we further bound the KL divergence by TV distance to obtain a proper metric bound for the final result. The full proof is given in Appendix B. Since



	Pendulum-v0	LunarLander-v2	Hopper-v2	Walker2d-v2	Ant-v2	CarRacing-v0
$\|\mathcal{S}\|$:	3-D	8-D	11-D	17-D	111-D	96×96 (image)
$\|\mathcal{A}\|$:	1-D	2-D	3-D	6-D	8-D	3-D
T:	100	250	1000	1000	1000	1000

Figure 2: Benchmark tasks. $|\mathcal{S}|$ and $|\mathcal{A}|$ denote the dimensions of the state space and the action space, respectively. T denotes the task horizon.

RHC performs local optimization in each iteration, we cannot guarantee global optimality. However, the theorem indicates that unlike standard behavior cloning, the cumulative error of RHIRL grows linearly and not quadratically in T. This shows one advantage of IRL over behavior cloning from the theoretical angle.

Given a control policy V with the resulting state marginal distribution $p_V(x)$, we can calculate the expected total cost of V by integrating the one-step cost over p_V. Now suppose that the one-step cost is bounded. Theorem 3.1 then implies that the regret in total cost, compared with the expert policy, also grows linearly in T.

3.6 Extension to Stochastic Dynamics

Suppose that the system dynamics is stochastic: $x_{t+1} = f(x_t, v_t, \omega_t)$, where ω_t is a random variable that models the independent system noise. RHIRL still applies, with modifications. We redefine the total cost functions $\tilde{J}(V, x_0; \theta)$ and $\tilde{S}(V, x_0; \theta)$ by taking expectation over the system noise distribution. When calculating the importance weight $\tilde{w}(V)$ in (11), we sample over the noise distribution to estimate the expected total state cost. Finally, we may need more samples when estimating the gradient in (12), because of the increased variance due to the system noise. The full derivation of the extension to stochastic dynamics is given in Appendix C. The experiments in the current work all have deterministic dynamics. We leave experiments with the extension to future work.

4 Experiments

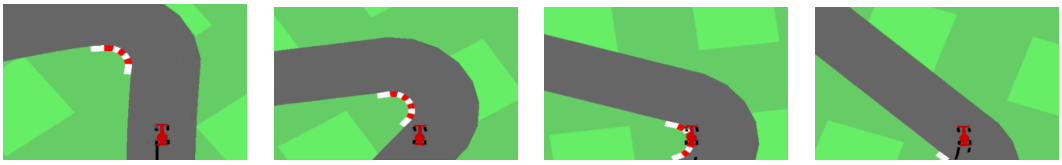
We investigate two main questions. Does RHIRL scale up to high-dimensional continuous control tasks? Does RHIRL learn a robust cost function under noise?

4.1 Setup

We compare RHIRL with two leading IRL algorithms, namely AIRL [8] and f-IRL [27], and one imitation learning algorithm, GAIL [13]. In particular, f-IRL is a recent algorithm that achieves leading performance on high-dimensional control tasks. We use the implementation of AIRL, GAIL, and f-IRL from the f-IRL's official repository along with the reported hyperparameters [27], whenever possible. We also perform hyperparameter search on a grid to optimize the performance of every method on every task. The specific hyperparameter settings used are reported in Appendix D.2.

Our benchmark set consists of six continuous control tasks (Figure 2) from OpenAI Gym [2], with increasing sizes of state and action spaces. For the most complex task, CarRacing, the input consists of 96×96 raw images, resulting in an enormous state space that poses great challenges [11]. To our knowledge, RHIRL is the first few IRL algorithms to attempt such a high-dimensional space. For fair comparison, we customize all tasks to the fixed task horizon settings (Figure 2. See Appendix D.2 for details on task parameter settings.

We use WorldModel [11] to generate expert demonstration data for CarRacing and use SAC [12] for the other tasks. We add Gaussian noise to the input controls and collect expert demonstrations at different control noise levels. The covariance of the control noise is *unknown* to all methods, including, in particular, RHIRL.



Drive in the center of the lane. Adjust the direction at a sharp turn. Choose the shortest path to make the turn. Align to the lane center after the turn.

Figure 3: Driving behaviors learned by RHIRL for CarRacing-v0, with pixel-level raw image input.

To measure the performance of the learned cost function and policy, we score its induced optimal policy using the ground-truth cost function. For ease of comparison with the literature, we use negated cost values, i.e., rewards, in all reported results. Higher values indicate better performance. Each experiment is repeated 10 times to estimate the performance variance.

4.2 Scalability

We compare RHIRL with other methods, first in noise-free environments (Figure 4) and then with increasing noise levels (Table 1).

Table 1: Performance comparison of RHIRL and other methods. The performance is reported as the ratio of the learned policy's average return and the expert's average return. The absolute average returns and the standard deviations are reported in Appendix D.4. Negative ratios are clipped to 0. The two numbers under the name of each environment indicate the dimensions of the state space and the action space, respectively.

		No Noise $\Sigma = 0$	**Mild Noise** $\Sigma = 0.2$	**High Noise** $\Sigma = 0.5$
Pendulum 3, 1	Expert	-154.69 $\pm$ 67.61	-156.50 $\pm$ 70.72	-168.54 $\pm$ 80.89
	RHIRL	1.06	**1.07**	**1.08**
	f-IRL	**1.07**	1.06	0.93
	AIRL	1.05	0.94	0.91
	GAIL	0.88	0.89	0.80
Lunarlander 8, 2	Expert	232.00 $\pm$ 86.12	222.65 $\pm$ 56.35	164.52 $\pm$ 16.79
	RHIRL	**1.05**	**1.04**	**1.20**
	f-IRL	0.76	0.63	0.74
	AIRL	0.74	0.60	0.58
	GAIL	0.72	0.56	0.60
Hopper 11, 3	Expert	3222.48 $\pm$ 390.65	3159.72 $\pm$ 520.00	2887.72 $\pm$ 483.93
	RHIRL	0.95	**0.98**	**0.96**
	f-IRL	**0.96**	0.82	0.43
	AIRL	0.01	0.01	0.01
	GAIL	0.82	0.50	0.24
Walker2d 17, 6	Expert	4999.47 $\pm$ 55.99	4500.43 $\pm$ 114.48	3624.48 $\pm$ 95.05
	RHIRL	**0.99**	**0.99**	**0.95**
	f-IRL	**0.99**	0.82	0.78
	AIRL	0.00	0.00	0.00
	GAIL	0.50	0.64	0.51
Ant 111, 8	Expert	5759.22 $\pm$ 173.57	3257.37 $\pm$ 501.95	252.62 $\pm$ 91.44
	RHIRL	0.86	**0.93**	**0.91**
	f-IRL	**0.87**	0.80	0.78
	AIRL	0.17	0.33	0.00
	GAIL	0.48	0.40	0.00
CarRacing 96×96, 3	Expert	903.25 $\pm$ 0.23	702.01 $\pm$ 0.3	281.12 $\pm$ 0.34
	RHIRL	**0.40**	**0.29**	**0.19**
	f-IRL	0.09	0.03	0.00
	AIRL	0.00	0.00	0.00
	GAIL	0.00	0.01	0.00

Figure 4 shows the learning curve of each methods in noise-free environments. Overall, RHIRL converges faster and achieves higher return, especially for tasks with higher state space dimensions. This improved performance suggests that the benefit of local optimization adopted by RHIRL outweighs its potential limitations.

Table 1 shows the final performance of all methods at various noise levels. RHIRL clearly outperforms AIRL and GAIL in all experiments. So we focus our discussion on comparison with f-IRL. In noise-free environments, RHIRL and f-IRL perform comparably on most tasks. On CarRacing, the most challenging task, RHIRL performs much better. RHIRL manages to learn the critical driving behaviors illustrated in Figure 3, despite the high-dimensional image input. However, RHIRL does not manage to learn to drive fast enough. That is the main reason why it under-performs the expert. In comparison, f-IRL only learns to follow a straight lane after a large number of environment steps, and still fails to make a sharp turn after 3.0×10^7 environment steps. In the noisy environments, the advantage of RHIRL over f-IRL is more pronounced even on some of the low-dimensional tasks, because RHIRL accounts for the control noise explicitly in the cost function.

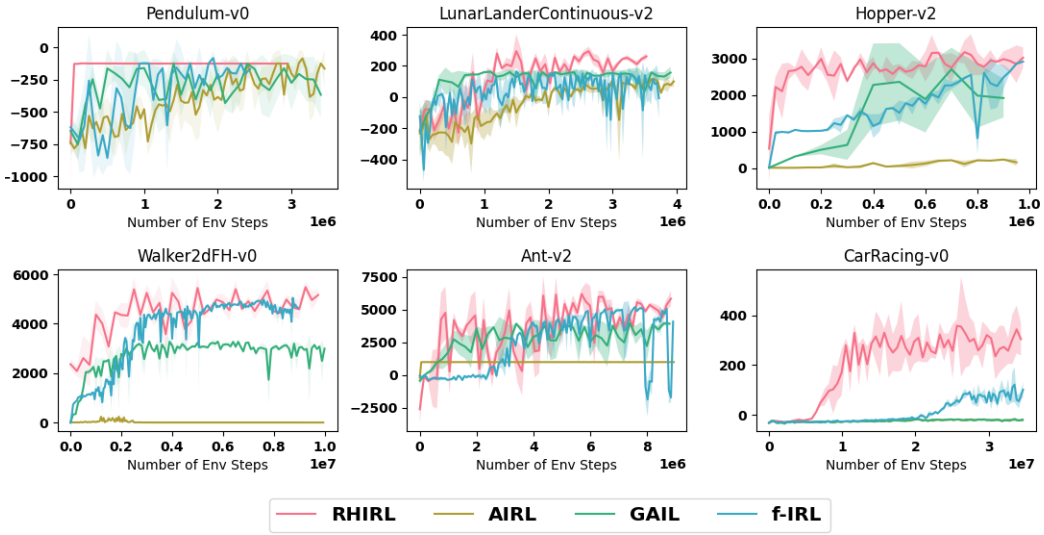


Figure 4: Learning curves for RHIRL and other methods.

4.3 Robustness

Next we evaluate RHIRL and other methods for robustness under noise. A robust cost function encodes the expert's true intent. It is expected to yield consistent performance over different noise levels, regardless of noise in expert demonstrations.

For each task, a cost function is learned in a noise-free environment and is then used to re-optimize a policy in the corresponding noisy environments. Specifically for GAIL, since it learns a policy and does not recover the associated cost function, we directly apply the learned policy in noisy environments.

Table 2 shows that noise causes performance degradation in all methods. However, RHIRL is more robust in comparison. For simple tasks, Pendulum and Lunarlander, RHIRL and f-IRL perform consistently well across different noise levels, while GAIL and AIRL fail to maintain their good performance, when the noise level increases. For the more challenging tasks, Hopper and Walker, RHIRL's performance degrades mildly, and f-IRL suffers more significant performance degradation. It is worth noting that the expert demonstrations used to train the transferred cost function are from the perfect system. Therefore, some expert actions and states may no longer be optimal or feasible in a highly noisy environment.

Table 2: Robustness of RHIRL and other methods under noise. The performance is measured as the ratio between the average return of an re-optimized policy in a noisy environment and the expert's average return in the corresponding noise-free environment. The absolute average returns and the standard deviations are reported in Appendix D.4. The negative ratios are clipped to 0.

		Noise Level Σ		
		0.0	0.2	0.5
Pendulum	Expert	-154.69 $\pm$ 67.61	–	–
3, 1	RHIRL	1.06	**1.07**	**1.06**
	f-IRL	**1.08**	0.90	0.85
	AIRL	1.05	0.79	0.67
	GAIL	0.88	0.71	0.62
LunarLander	Expert	232.00 $\pm$ 86.12	–	–
8, 2	RHIRL	**1.05**	**0.89**	**0.76**
	f-IRL	0.76	0.53	0.44
	AIRL	0.74	0.14	0.10
	GAIL	0.72	0.44	0.34
Hopper	Expert	3222.48 $\pm$ 390.65	–	–
11, 3	RHIRL	0.95	**0.80**	**0.67**
	f-IRL	**0.96**	0.65	0.62
	AIRL	0.01	0.01	0.00
	GAIL	0.82	0.07	0.06
Walker	Expert	4999.47 $\pm$ 55.99	–	–
17, 6	RHIRL	**0.99**	**0.80**	**0.69**
	f-IRL	**0.99**	0.60	0.22
	AIRL	0.00	0.28	0.36
	GAIL	0.50	0.02	0.02
Ant	Expert	5759.22 $\pm$ 173.57	–	–
111, 8	RHIRL	**0.86**	**0.55**	**0.15**
	f-IRL	0.87	0.35	0.08
	AIRL	0.17	0.15	0.00
	GAIL	0.48	0.00	0.00
CarRacing	Expert	903.25 $\pm$ 0.23	–	–
96×96, 3	RHIRL	**0.40**	**0.29**	**0.12**
	f-IRL	0.09	0.02	0.00
	AIRL	0.00	–	–
	GAIL	0.00	–	–

Moreover, the cost function trained in the perfect system cannot reason about the long-term consequences of an action in a high noise environment. Therefore, it is challenging for the learned cost function to be robust to a highly noisy environment, as capturing the true intention of the expert is difficult.

4.4 Effect of Receding Horizon K

RHIRL uses the receding horizon K to trade off optimality and efficiency. We hope to ablate the effect of K on Hopper-v2 to show how different Ks affect the final performance and sample complexity. The task horizon for Hopper-v2 is 1000 steps, i.e. $T = 1000$. We run RHIRL with the receding horizon $K \in \{5, 20, 100\}$. The results are illustrated in Figure 5. When K is small, RHIRL improves its performance quickly but converges to the suboptimal solution. For $K = 5$, RHIRL's performance shoots up after the first few iterations to 1000, then it quickly converges to a final score of 1100. When K increases, though the performance improves slightly slower than $K = 5$, it can continue to learn and reach a score of 3071.68. At $K = 20$, it takes fewer than $1e6$ env steps to stabilize to a score greater than 3000. However, when K is too large, the learning becomes much slower. When $K = 100$, it takes more than $1e7$ env steps to stabilize to a score larger than 3000, which is 10 times more than when $K = 20$. On the other hand, $K = 100$ can achieve a final score of 3083, which is slightly more than that of $K = 20$. This ablation study shows that our receding horizon K can tradeoff optimality and efficiency: using a smaller K allows us to learn faster at the expense of a sub-optimal solution, while using a large K may make the learning inefficient. Seeking a suitable K can balance the requirement for optimality and efficiency.

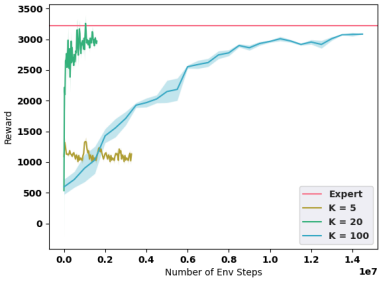


Figure 5: The effect of receding horizon K on the performance of RHIRL on the Hopper-v2 task.

5 Conclusion

RHIRL is a scalable and robust IRL algorithm for high-dimensional, noisy, continuous systems. Our experiments show that RHIRL outperforms several leading IRL algorithms on multiple benchmark tasks, especially when expert demonstrations are noisy.

RHIRL's choice of local rather than global optimization is an important issue that deserves further investigation. Overall, we view this as an interesting trade-off between scalability and optimality. While this trade-off is well known in reinforcement learning, optimal control, and general optimization problems, it is mostly unexplored in IRL. Further, local optimization may tie the learned cost with the optimizer. It would be interesting to examine whether the learned cost transfers easily to other domains with different optimizers. We are keen to investigate these important issues and their implications to IRL as our next step.

Acknowledgments This research is supported in part by the Agency of Science, Technology and Research, Singapore, through the National Robotics Program (Grant No. 192 25 00054), the National Research Foundation (NRF), Singapore and DSO National Laboratories under the AI Singapore Program (AISG Award No: AISG2-RP-2020-016), and the National University of Singapore (AcRF grant A-0008071-00-00). Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not reflect the views of NRF Singapore.

References

[1] A. Boularias, J. Kober, and J. Peters. Relative entropy inverse reinforcement learning. In *Proc. Int. Conf. on Artificial Intelligence & Statistics*, 2011.

[2] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

[3] D. S. Brown, W. Goo, P. Nagarajan, and S. Niekum. Extrapolating beyond suboptimal demonstrations via inverse reinforcement learning from observations. In *Proc. Int. Conf. on Machine Learning*, 2019.

[4] D. S. Brown, W. Goo, and S. Niekum. Better-than-demonstrator imitation learning via automatically-ranked demonstrations. In *Proc. Conference on Robot Learning*, 2019.

[5] L. Chen, R. R. Paleja, and M. C. Gombolay. Learning from suboptimal demonstration via self-supervised reward regression. *Proc. Conference on Robot Learning*, 2020.

[6] C. Finn, P. Christiano, P. Abbeel, and S. Levine. A connection between generative adversarial networks, inverse reinforcement learning, and energy-based models. In *Advances in Neural Information Processing Systems*, 2016.

[7] C. Finn, S. Levine, and P. Abbeel. Guided cost learning: Deep inverse optimal control via policy optimization. In *Proc. Int. Conf. on Machine Learning*, 2016.

[8] J. Fu, K. Luo, and S. Levine. Learning robust rewards with adversarial inverse reinforcement learning. In *Proc. Int. Conf. on Learning Representations*, 2018.

[9] S. K. S. Ghasemipour, R. Zemel, and S. Gu. A divergence minimization perspective on imitation learning methods. In *Proc. Conference on Robot Learning*, 2019.

[10] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial networks. In *Advances in Neural Information Processing Systems*, 2014.

[11] D. Ha and J. Schmidhuber. World models. In *Advances in Neural Information Processing Systems*, 2018.

[12] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *Proc. Int. Conf. on Machine Learning*, 2018.

[13] J. Ho and S. Ermon. Generative adversarial imitation learning. In *Advances in Neural Information Processing Systems*, 2016.

[14] H. Hoshino, K. Ota, A. Kanezaki, and R. Yokota. Opirl: Sample efficient off-policy inverse reinforcement learning via distribution matching. In *Proc. IEEE Int. Conf. on Robotics & Automation*, 2021.

[15] D. Kahneman and A. Tversky. Prospect theory: An analysis of decision under risk. *Econometrica*, 47(2):263–291, 1979.

[16] L. Ke, S. Choudhury, M. Barnes, W. Sun, G. Lee, and S. Srinivasa. Imitation learning as f-divergence minimization. In *Algorithmic Foundations of Robotics XIV—Proc. Int. Workshop on the Algorithmic Foundations of Robotics (WAFR)*, 2020.

[17] W. B. Knox, A. Allievi, H. Banzhaf, F. Schmitt, and P. Stone. Reward (mis)design for autonomous driving. *arXiv preprint arXiv:2104.13906*, 2021.

[18] I. Kostrikov, K. K. Agrawal, D. Dwibedi, S. Levine, and J. Tompson. Discriminator-actor-critic: Addressing sample inefficiency and reward bias in adversarial imitation learning. In *Proc. Int. Conf. on Learning Representations*, 2019.

[19] W.-H. Kwon and S. Han. *Receding Horizon Control: Model Predictive Control for State Models*. Springer-Verlag London, 2005.

[20] K. Lee, D. Isele, E. A. Theodorou, and S. Bae. Spatiotemporal costmap inference for MPC via deep inverse reinforcement learning. In *IEEE Robotics & Automation Letters*, 2022.

[21] L. Lee, B. Eysenbach, E. Parisotto, E. P. Xing, S. Levine, and R. Salakhutdinov. Efficient exploration via state marginal matching. *ArXiv*, abs/1906.05274, 2019.

[22] S. Levine and V. Koltun. Guided policy search. In *Proc. Int. Conf. on Machine Learning*, 2013.

[23] J. MacGlashan and M. L. Littman. Between imitation and intention learning. In *Proc. Int. Jnt. Conf. on Artificial Intelligence*, 2015.

[24] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. A. Riedmiller. Playing atari with deep reinforcement learning. In *Advances in Neural Information Processing Systems*, 2013.

[25] A. Ng, D. Harada, and S. Russell. Policy invariance under reward transformations: Theory and application to reward shaping. In *Proc. Int. Conf. on Machine Learning*, 1999.

[26] A. Y. Ng and S. J. Russell. Algorithms for inverse reinforcement learning. In *Proc. Int. Conf. on Machine Learning*, 2000.

[27] T. Ni, H. Sikchi, Y. Wang, T. Gupta, L. Lee, and B. Eysenbach. f-irl: Inverse reinforcement learning via state marginal matching. In *Proc. Conference on Robot Learning*, 11 2020.

[28] W. Powell. *Approximate Dynamic Programming: Solving the Curses of Dimensionality*. Wiley-Interscience, 2007.

[29] A. H. Qureshi, B. Boots, and M. C. Yip. Adversarial imitation via variational inverse reinforcement learning. In *Proc. Int. Conf. on Learning Representations*, 2019.

[30] S. Ross, G. J. Gordon, and J. A. Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proc. Int. Conf. on Artificial Intelligence & Statistics*, 2011.

[31] K. Shiarlis, J. V. Messias, and S. Whiteson. Inverse reinforcement learning from failure. In *Proc. Int. Conf. on Autonomous Agents & Multiagent Systems*, 2016.

[32] J. Spencer, S. Choudhury, A. Venkatraman, B. Ziebart, and J. A. Bagnell. Feedback in imitation learning: The three regimes of covariate shift, 2021.

[33] E. A. Theodorou and E. Todorov. Relative entropy and free energy dualities: Connections to path integral and kl control. In *Proc. IEEE Conference on Decision & Control*, 2012.

[34] E. Todorov and M. Jordan. Optimal feedback control as a theory of motor coordination. *Nature neuroscience*, 5:1226–35, 12 2002.

[35] U. Viereck, A. Pas, K. Saenko, and R. Platt. Learning a visuomotor controller for real world robotic grasping using simulated depth images. In *Proc. Conference on Robot Learning*, 2017.

[36] G. Williams, B. Goldfain, P. Drews, K. Saigol, J. M. Rehg, and E. A. Theodorou. Robust sampling based model predictive control with sparse objective information. *Robotics: Science and Systems XIV*, 2018.

[37] G. Williams, N. Wagener, B. Goldfain, P. Drews, J. M. Rehg, B. Boots, and E. A. Theodorou. Information theoretic mpc for model-based reinforcement learning. In *Proc. IEEE Int. Conf. on Robotics & Automation*, 2017.

[38] Y. H. Wu, N. Charoenphakdee, H. Bao, V. Tangkaratt, and M. Sugiyama. Imitation learning from imperfect demonstration. In *Proc. Int. Conf. on Machine Learning*, 2019.

[39] M. Wulfmeier, P. Ondruska, and I. Posner. Maximum entropy deep inverse reinforcement learning. In *Advances in Neural Information Processing Systems*, 2016.

[40] B. D. Ziebart, J. A. Bagnell, and A. K. Dey. Modeling interaction via the principle of maximum causal entropy. In *Proc. Int. Conf. on Machine Learning*, 2010.

[41] B. D. Ziebart, A. Maas, J. A. Bagnell, and A. K. Dey. Maximum entropy inverse reinforcement learning. In *Proc. AAAI Conf. on Artificial Intelligence*, 2008.

Checklist

The checklist follows the references. Please read the checklist guidelines carefully for information on how to answer these questions. For each question, change the default **[TODO]** to [Yes] , [No] , or [N/A] . You are strongly encouraged to include a **justification to your answer**, either by referencing the appropriate section of your paper or providing a brief inline description. For example:

- Did you include the license to the code and datasets? [Yes] See Section **??**.
- Did you include the license to the code and datasets? [No] The code and the data are proprietary.
- Did you include the license to the code and datasets? [N/A]

Please do not modify the questions and only use the provided macros for your answers. Note that the Checklist section does not count towards the page limit. In your paper, please delete this instructions block and only keep the Checklist section heading above along with the questions/answers below.

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] See Section 5
   (c) Did you discuss any potential negative societal impacts of your work? [N/A]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes]
   (b) Did you include complete proofs of all theoretical results? [Yes] See Appendix
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] , code is included in the supplementary material.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes]
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [N/A]
   (b) Did you mention the license of the assets? [N/A]
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

A Gradient Derivation

This section derives the gradient $\frac{\partial \mathcal{L}}{\partial \theta}$ in equation (12) in full details. Recall $\mathcal{L}$ is defined as:

$$\mathcal{L} = \int p_E(V) \log \frac{p_E(V)}{p_Q(V|U_B, \Sigma; \theta)} dV$$

Firstly, we substitute $\mathcal{L}$ in $\frac{\partial \mathcal{L}}{\partial \theta}$ and rewrite $\log \frac{p_E(V)}{p_Q(V|U_B,\Sigma;\theta)}$ as the difference between $\log\ p_E(V)$ and $p_Q(V|U_B, \Sigma; \theta)$.

$$\begin{aligned}
\frac{\partial \mathcal{L}}{\partial \theta} &= \frac{\partial}{\partial \theta} \int p_E(V) \log \frac{p_E(V)}{p_Q(V|U_B, \Sigma; \theta)} dV \\
&= \int p_E(V) \frac{\partial}{\partial \theta} \log\ p_E(V) dV - \int p_E(V) \frac{\partial}{\partial \theta} \log\ p_Q(V|U_B, \Sigma; \theta) dV \\
&= - \int p_E(V) \frac{\partial}{\partial \theta} \log\ p_Q(V|U_B, \Sigma; \theta) dV
\end{aligned} \tag{14}$$

Since $p_E(V)$ is independent of θ, the first derivative in the second line of equation (14) evaluates to 0. Next, we substitute $p_Q(V|U_B, \Sigma; \theta)$ using the optimal control sequence distribution expression in equation (4):

$$\begin{aligned}
\frac{\partial \mathcal{L}}{\partial \theta} &= - \int p_E(V) \frac{\partial}{\partial \theta} \log\ \frac{1}{Z} p_B(V|U_B, \Sigma) \exp(-\frac{1}{\lambda} S(V; \theta)) dV \\
&= - \int p_E(V) \frac{\partial}{\partial \theta} \log p_B(V|U_B, \Sigma)\ dV - \int p_E(V) \frac{\partial}{\partial \theta} \log \exp(-\frac{1}{\lambda} S(V; \theta)) dV \\
&\quad + \int p_E(V) \frac{\partial}{\partial \theta} \log Z dV \\
&= - \int p_E(V) \frac{\partial}{\partial \theta} (-\frac{1}{\lambda} S(V; \theta)) dV + \int p_E(V) \frac{\partial}{\partial \theta} \log Z dV
\end{aligned} \tag{15}$$

Since $p_B(V|U_B, \Sigma)$ is independent of θ, the first derivative in the second line of equation (14) evaluates to 0. We are left with only two integrals in equation (15).

Next, we factorize out $\frac{\partial}{\partial \theta} \log Z$ from the integral since the partition function Z is constant to all V:

$$\begin{aligned}
\frac{\partial \mathcal{L}}{\partial \theta} &= - \int p_E(V) (-\frac{1}{\lambda} \frac{\partial}{\partial \theta} S(V; \theta)) dV + (\frac{\partial}{\partial \theta} \log Z) \int p_E(V) dV \\
&= - \int p_E(V) (-\frac{1}{\lambda} \frac{\partial}{\partial \theta} S(V; \theta)) dV + \frac{1}{Z} \frac{\partial Z}{\partial \theta}
\end{aligned} \tag{16}$$

The second line in equation (16) follows as $\int p_E(V) dV = 1$.

Next, we substitute $Z = \int p_B(V|U_B, \Sigma) \exp(-\frac{1}{\lambda} S(V; \theta)) dV$ in $\frac{\partial Z}{\partial \theta}$ and simplify it:

$$\begin{aligned}
\frac{\partial \mathcal{L}}{\partial \theta} &= - \int p_E(V) (-\frac{1}{\lambda} \frac{\partial}{\partial \theta} S(V; \theta)) dV + \frac{1}{Z} \frac{\partial}{\partial \theta} \int p_B(V|U_B, \Sigma) \exp(-\frac{1}{\lambda} S(V; \theta)) dV \\
&= - \int p_E(V) (-\frac{1}{\lambda} \frac{\partial}{\partial \theta} S(V; \theta)) dV + \frac{1}{Z} \int p_B(V|U_B, \Sigma) \frac{\partial}{\partial \theta} \exp(-\frac{1}{\lambda} S(V; \theta)) dV \\
&= - \int p_E(V) (-\frac{1}{\lambda} \frac{\partial}{\partial \theta} S(V; \theta)) dV \\
&\quad + \frac{1}{Z} \int p_B(V|U_B, \Sigma) \exp(-\frac{1}{\lambda} S(V; \theta)) \frac{\partial}{\partial \theta} (-\frac{1}{\lambda} S(V; \theta)) dV \\
&= - \int p_E(V) (-\frac{1}{\lambda} \frac{\partial}{\partial \theta} S(V; \theta)) dV \\
&\quad + \int \frac{p_B(V|U_B, \Sigma) \exp(-\frac{1}{\lambda} S(V; \theta))}{Z} \frac{\partial}{\partial \theta} (-\frac{1}{\lambda} S(V; \theta)) dV
\end{aligned} \tag{17}$$

We rewrite $\frac{1}{Z}p_B(V|U_B,\Sigma)\exp(-\frac{1}{\lambda}S(V;\theta))$ in the last line of equation (17) as $p_Q(V|U_B,\Sigma;\theta)$ using equation (4) and finally we have:

$$\frac{\partial\mathcal{L}}{\partial\theta} = \int p_E(V)(\frac{1}{\lambda}\frac{\partial}{\partial\theta}S(V;\theta))dV - \int p_Q(V|U_B,\Sigma;\theta)(\frac{1}{\lambda}\frac{\partial}{\partial\theta}S(V;\theta))dV \tag{18}$$

B Theorem 3.1 and Proof

We use T to denote the task horizon, and K to denote the receding time horizon for local optimization. Let us simplify notations in the optimal control sequence distribution $p_Q(V|U_B,\Sigma,x_t;\theta)$ and remove the explicit dependency on U_B and Σ. We assume that all control sequences are applied with U_B as the base distribution and Σ as the covariate matrix. We assume the expert's underlying cost function is parameterized by θ_E, so we have $p_E(.) = p(.;\theta_E)$.

B.1 Sketch

We present a theoretical analysis on the convergence of RHIRL. Our main theorem 3.1 states that given that the Kullback–Leibler (KL) divergence over local control sequence distribution for each time step $t = 0, 1, ...T-1$ is bounded by ϵ, though we do not query expert during the learning, using the cached expert demonstrations alone allows us to bound the error over global state marginal distribution *linear* in the task horizon T under total variance measure.

First, we show that if the KL-divergence over the local control sequence distribution is bounded by ϵ, so is the KL-divergence over the resulting state distribution. At each time step t, the optimal control sequences distribution $p_Q(V_t|x_t;\theta_t)$ at the initial state x_t contains the full information to generate the corresponding state trajectories $p(\tau_t|x_t;\theta)$, and consequently the state distributions $p_t(x|x_t;\theta)$ (by neglecting the temporal information). Upon applying the information loss (lemma B.1), we prove in lemma B.2 that, given initial state x_t, if the KL-divergence over V_t is bound by ϵ, so is the corresponding state distribution $p_t(x)$, i.e. $D_{\text{KL}}(p_t(x|x_t;\theta_E) \parallel p_t(x|x_t;\theta)) \leq D_{\text{KL}}(p(V_t|x_t;\theta_E) \parallel p_Q(V_t|x_t;\theta)) \leq \epsilon$.

Next, we show the KL-divergence over global state marginal distribution between two consecutive time steps is also bounded by ϵ. For each optimal control sequence we compute, we only execute the first control and use the rest to warm start the re-planning for the next time step. Therefore, for each time step, we only change a small region of the global state distribution, i.e. reachable space of the current time step. We use $p^t_{\text{RHC}}(x;\theta)$ to denote the global state marginal distribution by recursively applying RHC from time step $i = 0, ..., t$ under θ and switching to θ_E thereafter until $T-1$. Using generalized log sum inequality, we prove in lemma B.3 that if the KL-divergence over V_t is bounded by ϵ for all $t = 0, ..., T-1$, the KL-divergence over the global state marginal distribution between each of the two consecutive time steps is bounded by ϵ, i.e. $D_{\text{KL}}(p^t_{\text{RHC}}(x;\theta) \parallel p^{t+1}_{\text{RHC}}(x;\theta)) \leq \frac{K+1}{T}\epsilon$.

Finally, we use Pinsker's inequality to upper bound the total variation (TV) distance by KL-divergence over state marginal distribution. Then we use the triangle inequality to show that the TV distance between expert and the actual visited state distribution over the task horizon T using RHIRL is bounded by an error linearly in T, i.e. $D_{\text{TV}}(p_E(x) \parallel p_{\text{RHC}}(x;\theta)) < T\sqrt{\epsilon/2}$.

B.2 Proofs

First, we use **Lemma B.1, B.2** to prove that the KL-divergence over control sequence space upper bounds the KL-divergence over the resulting state distribution. We define the control sequence starts at task time step t as $V_t = \{v_t, v_{t+1}, ..., v_{t+K-1}\}$. Moreover, its corresponding trajectory segment $\tau_t = \{x_t, x_{t+1}, ..., x_{t+K}\}$ is computed uniquely from V_t and initial state x_t by iteratively applying the dynamic model $x_{t+1} = f(x_t, v_t)$. We use $p(x_t)$ to denote the state density at a single time step t. Assume V_t is optimized based on the cost parameterization θ, then the corresponding state distribution is defined as the summation of all state density over horizon K, i.e. $p_t(x;\theta) = \frac{1}{K+1}\sum_{i=t}^{t+K} p(x_i;\theta)$.

Lemma B.1 (Information loss[16])**.** *Let a and b be two random variables and $f(.)$ be a convex function. Let $P(a,b)$ be a joint probability distribution. The marginal distributions are $P(a) = \sum_b P(a,b)$ and $P(b) = \sum_a P(a,b)$. Assume that a can explain away b. This is expressed as follows – given any two probability distribution $P(.)$, $Q(.)$, assume the following equality holds for all a, b:*

$$P(b|a) = Q(b|a) \tag{19}$$

Under these conditions, the following inequality holds:

$$\sum_a Q(a) f(\frac{P(a)}{Q(a)}) > \sum_b Q(b) f(\frac{P(b)}{Q(b)}) \tag{20}$$

Lemma B.2. *Given the initial state x_t and the two control sequence distributions $p(V_t|x_t;\theta_E)$ and $p_Q(V_t|x_t;\theta)$, the KL-divergence between the resulting state distribution is upper bounded by KL-divergence between the control sequence distribution.*

$$D_{KL}(p_t(x|x_t;\theta_E) \parallel p_t(x|x_t;\theta)) \leq D_{KL}(p(V_t|x_t;\theta_E) \parallel p_Q(V_t|x_t;\theta)) \tag{21}$$

Proof. Firstly, we prove that the KL-divergence over state trajectory distribution $p(\tau_t)$ is upper bounded by the KL-divergence between $p(V_t)$. Given dynamical model f, the control sequence V_t and the initial state x_t contains all information to generate the corresponding τ_t. Therefore, for any joint distribution $P(\tau_t, V_t|x_t)$ and $Q(\tau_t, V_t|x_t)$, the following is true

$$P(\tau_t|V_t, x_t) = Q(\tau_t|V_t, x_t)$$

Upon applying the information loss Lemma B.1, we have the inequality:

$$D_{\text{KL}}(p(\tau_t|x_t;\theta_E) \parallel p(\tau_t|x_t;\theta)) \leq D_{\text{KL}}(p(V_t|x_t;\theta_E) \parallel p_Q(V_t|x_t;\theta)) \tag{22}$$

Next we prove that the KL-divergence between state distribution is upper bounded by the trajectory distribution. Since a trajectory $\tau_t = \{x_t, x_{t+1}, ..., x_{t+K}\}$ contains full information of the resulting states (by neglecting the temporal information), for any joint distribution $P(x|\tau_t)$ and $Q(x|\tau_t)$, the following is true

$$P(x|\tau_t) = Q(x|\tau_t) \tag{23}$$

Upon applying Lemma B.1 we have the inequality:

$$D_{\text{KL}}(p_t(x|x_t;\theta_E) \parallel p_t(x|x_t;\theta)) \leq D_{\text{KL}}(p(\tau_t|x_t;\theta_E) \parallel p(\tau_t|x_t;\theta)) \tag{24}$$

Therefore, given the KL-divergence between the control sequence distribution is upper bounded by ϵ, we use the equality in equation (22) and (24) to show that the KL-divergence between the resulting state distribution is also upper bounded by ϵ.

$$D_{\text{KL}}(p_t(x|x_t;\theta_E) \parallel p_t(x|x_t;\theta)) \leq D_{\text{KL}}(p(V_t|x_t;\theta_E) \parallel p_Q(V_t|x_t;\theta)) \leq \epsilon \tag{25}$$

□

Definition 1 (One-step Recoverability [32])**.** Assume that the state distribution of the learner and expert are different at time t, that is $D_{\text{KL}}(p(x_t;\theta_E) \parallel p(x_t;\theta)) \neq 0$, there exists a policy π_{re} that when used for the learner, can bound:

$$D_{\text{KL}}(p(x_{t+1};\theta_E) \parallel p(x_{t+1};\pi_{re})) \leq \epsilon_1 \tag{26}$$

where the current initial state distribution of the student follows $p(x_t;\theta)$.

Intuitively, this condition requires that, no matter what is the current state distribution, the learner can recover to the expert demonstrated distribution in a single time-step. In our case, this is a natural condition since the difference in the initial state distribution $p(x_t;\theta)$ and $p(x_t;\theta_E)$ is not arbitrarily large: we use re-planning to ensure the receding state sequences is always bounded below ϵ, hence this recoverability condition can be easily satisfied. We emphasize that this recoverable policy is **never** executed in our algorithm, it is only used for the theoretical analysis.

Next, we derive a bound over the global state marginal distribution between two consecutive time steps. At each time step $t = 0, ..., T-1$, we re-optimize the local control sequence distribution and only execute the first control, hence we only change state density over a small reachable space. We define $p^t_{\text{RHC}}(x,\theta)$ as the global marginal state distribution by applying RHC from $i = 0, 1, ..., t$ under θ and then using the recoverable policy π_{re} to switch to the expert θ_E thereafter until $T-1$, i.e., $p^t_{\text{RHC}}(x;\theta) = \frac{1}{T}(\sum_{i=0}^{t+1} p(x_i;\theta) + p(x_{t+2};\pi_{re}) + \sum_{i=t+3}^{T} p(x_i;\theta_E))$. According to the definition of the recoverable policy, we have $D_{\text{KL}}(p(x_{t+2};\theta_E) \parallel p(x_{t+2};\pi_{re})) \leq \epsilon_1$, therefore, $p^t_{\text{RHC}}(x;\theta) \approx \frac{1}{T}(\sum_{i=0}^{t+1} p(x_i;\theta) + \sum_{i=t+2}^{T} p(x_i;\theta_E))$. To quantify the change in global state marginal distribution, we derive a bound for the KL-divergence between two consecutive time steps, i.e. $D_{\text{KL}}(p^{t-1}_{\text{RHC}}(x;\theta) \parallel p^t_{\text{RHC}}(x;\theta))$.

Lemma B.3. *If the KL-divergence over resulting state density from the control sequence distribution of length K are bounded by ϵ, i.e. $D_{\text{KL}}(p_t(x|x_t;\theta_E) \parallel p_t(x|x_t;\theta)) < \epsilon$, where x_t is the state encountered by our policy at $t = 0, 1, ..., T-1$ and is one-step recoverable, then KL-divergence over the global state marginal distribution between two consecutive control executions are bounded by ϵ,*

$$D_{KL}(p^{t-1}_{\text{RHC}}(x;\theta) \parallel p^{t}_{\text{RHC}}(x;\theta)) \leq \frac{K+1}{T}\epsilon \tag{27}$$

for $t = 1, ..., T-1$.

We state the generalized log sum inequality below in lemma B.4, the proof can be found in the Appendix of [16]. Lemma B.4 and B.5 will be used in the proof for Lemma B.3.

Lemma B.4 (Generalized log sum inequality[16])**.** *Let $p_1, ..., p_n$ and $q_1, ..., q_n$ be non-negative numbers. Let $p = \sum_{i=1}^{n} p_i$ and $q = \sum_{i=1}^{n} q_i$. Let $f(.)$ be a convex function. We have the following:*

$$\sum_{i=1}^{n} q_i f(\frac{p_i}{q_i}) \geq q f(\frac{p}{q}) \tag{28}$$

Lemma B.5. *Let $p(x)$ and $q(x)$ be non-negative functions, and c is a constant factor. We have the following:*

$$\int cp(x) \log \frac{cp(x)}{cq(x)} = c \int p(x) \log \frac{p(x)}{q(x)} \tag{29}$$

Proof.

$$\int cp(x) \log \frac{cp(x)}{cq(x)} dx = \int cp(x) \log \frac{p(x)}{q(x)} dx = c \int p(x) \log \frac{p(x)}{q(x)} dx \tag{30}$$

□

Now, we are ready to prove lemma B.3.

Proof. For each $t = 0, ..., T-1$, we re-plan for the optimal local control sequence start at x_t so that the resulting state distribution over horizon K is bounded, i.e. $D_{\text{KL}}(p_t(x|x_t;\theta_E) \parallel p_t(x|x_t;\theta)) < \epsilon$, where $p_t(x|x_t;\theta) = \frac{1}{K+1}\sum_{i=t}^{t+K} p(x_i;\theta)$. However, instead of executing all K controls in the sequence, we only execute the first control at the current time step t and change the state distribution $p(x_{t+1};\theta)$ reachable for that single time step, then we use the remaining control sequence to warm start the local control sequence optimization for the next time step. To account the effect of replanning, for each time step t, since we do not change the state distribution after $p(x_{t+1};\theta)$, we can think of the change in the global state distribution as if we follow the optimal control under θ at time step t and then use the recoverable policy π_{re} to switch to θ_E afterwards over the control sequence horizon K. Hence, the actual state density is $\frac{1}{K+1}(p(x_t;\theta) + p(x_{t+1};\theta) + p(x_{t+2};\pi_{re}) + \sum_{i=t+3}^{t+K} p(x_i;\theta_E))$. Theoretically, since we do not query the expert online, the initial state x_t distribution in theorem 3.1 and lemma B.2 should follow the expert demonstration at time t, i.e. $p(x_t;\theta_E)$. However, our MPC controller cannot jump to this distribution and we replan from our current state distribution $p(x_t;\theta)$. To resolve this mismatch, we require the recoverability condition in our optimization procedure such that, for each resulting state from the controller, there always exists a one-step recoverable policy π_{re} that can correct the current state distribution $p(x_t;\theta)$ to $p(x_{t+1};\theta_E)$ in one step. Therefore, with this one-step recoverable condition on every state x_t induced by the cost function parameterized by θ, x_t in theorem 3.1 and lemma B.2 now follows the state distribution of our controller, i.e. $p(x_t;\theta)$.

That is,

$$\begin{aligned}
&D_{\text{KL}}(p_t(x|x_t;\theta_E) \parallel \frac{1}{K+1}(p(x_t) + p(x_{t+1};\theta) + \sum_{i=t+2}^{t+K} p(x_i;\theta_E)) \\
=&D_{\text{KL}}(\frac{1}{K+1}(p(x_t;\theta) + p(x_{t+1};\pi_{re}) + \sum_{i=t+2}^{t+K} p(x;\theta_E)) \parallel \frac{1}{K+1}(p(x_t;\theta) + p(x_{t+1};\theta) + \sum_{i=t+2}^{t+K} p(x_i;\theta_E)) \\
\leq&D_{\text{KL}}(\frac{1}{K+1}(p(x_t;\theta) + \sum_{i=t+1}^{t+K} p(x;\theta_E)) \parallel \frac{1}{K+1}(p(x_t;\theta) + \sum_{i=t+1}^{t+K} p(x_i;\theta)) \\
=&D_{\text{KL}}(p_t(x|x_t;\theta_E) \parallel p_t(x|x_t;\theta)) \leq \epsilon
\end{aligned} \tag{31}$$

Recall that we use $p^t_{\text{RHC}}(x;\theta)$ to account for the global state marginal distribution resulted from executing a single optimal control at time step t. More specifically, $p^t_{\text{RHC}}(x;\theta)$ is defined as the state marginal distribution by executing only the first optimal control from the replanned optimal control sequence at each time step from $i = 0, ..., t$ under θ and switching to θ_E thereafter by using the recoverable policy π_{re} until $T-1$, i.e. $p^t_{\text{RHC}}(x;\theta) \approx \frac{1}{T}(\sum_{i=0}^{t+1} p(x_i;\theta) + \sum_{i=t+2}^{T} p(x_i;\theta_E))$. We bound the KL-divergence over global state distribution between two consecutive time step as follow:

$$
\begin{aligned}
&D_{\text{KL}}(p^{t-1}_{\text{RHC}}(x;\theta) \parallel p^t_{\text{RHC}}(x;\theta)) \\
=&D_{\text{KL}}(\frac{1}{T}(\sum_{i=0}^{t} p(x_i;\theta) + p(x_{t+1};\pi_{re}) + \sum_{i=t+2}^{T} p(x_i;\theta_E)) \parallel \frac{1}{T}(\sum_{i=0}^{t+1} p(x_i;\theta) + p(x_{t+2};\pi_{re}) + \sum_{i=t+3}^{T} p(x_i;\theta_E))) \\
\approx&D_{\text{KL}}(\frac{1}{T}(\sum_{i=0}^{t} p(x_i;\theta) + \sum_{i=t+1}^{T} p(x_i;\theta_E)) \parallel \frac{1}{T}(\sum_{i=0}^{t+1} p(x_i;\theta) + \sum_{i=t+2}^{T} p(x_i;\theta_E))) \\
=&\frac{1}{T}\int(\sum_{i=0}^{t} p(x_i;\theta) + \sum_{i=t+1}^{T} p(x_i;\theta_E)) \log \frac{\sum_{i=0}^{t} p(x_i;\theta) + \sum_{i=t+1} p(x_i;\theta_E)}{\sum_{i=0}^{t+1} p(x_i;\theta) + \sum_{i=t+2}^{T} p(x_i;\theta_E)} dx \\
\leq&\frac{1}{T}(\int \sum_{i=0}^{t-1} p(x_i;\theta) \log \frac{\sum_{i=0}^{t-1} p(x_i;\theta)}{\sum_{i=0}^{t-1} p(x_i;\theta)} dx \\
&+ \int (p(x_t;\theta) + \sum_{i=t+1}^{t+K} p(x_i;\theta_E)) \log \frac{p(x_t;\theta) + \sum_{i=t+1}^{t+K} p(x_i;\theta_E)}{p(x_t;\theta) + p(x_{t+1};\theta) + \sum_{i=t+2}^{t+K} p(x_i;\theta_E)} dx \\
&+ \int \sum_{i=t+K+1}^{T} p(x_i;\theta_E) \log \frac{\sum_{i=t+K+1}^{T} p(x_i;\theta_E)}{\sum_{i=t+K+1}^{T} p(x_i;\theta_E)} dx) \\
=&\frac{1}{T}\int (p(x_t;\theta) + \sum_{i=t+1}^{t+K} p(x_i;\theta_E)) \log \frac{p(x_t;\theta) + \sum_{i=t+1}^{t+K} p(x_i;\theta_E)}{p(x_t;\theta) + p(x_{t+1};\theta) + \sum_{i=t+2}^{t+K} p(x_i;\theta_E)} dx \\
=&\frac{K+1}{T}\int \frac{1}{K+1}(p(x_t;\theta) + \sum_{i=t+1}^{t+K} p(x_i;\theta_E)) \log \frac{\frac{1}{K+1}(p(x_t;\theta) + \sum_{i=t+1}^{t+K} p(x_i;\theta_E))}{\frac{1}{K+1}(p(x_t;\theta) + p(x_{t+1};\theta) + \sum_{i=t+2}^{t+K} p(x_i;\theta_E))} dx \\
=&\frac{K+1}{T} D_{\text{KL}}(\frac{1}{K+1}(p(x_t;\theta) + \sum_{i=t+1}^{t+K} p(x_i;\theta_E)) \parallel \frac{1}{K+1}(p(x_t;\theta) + p(x_{t+1};\theta) + \sum_{i=t+2}^{t+K} p(x_i;\theta_E))) \\
=&\frac{K+1}{T} D_{\text{KL}}(p_t(x|x_t;\theta_E) \parallel \frac{1}{K+1}(p(x_t;\theta) + p(x_{t+1};\theta) + \sum_{i=t+2}^{t+K} p(x_i;\theta_E))) \\
\leq&\frac{K+1}{T}\epsilon
\end{aligned}
\tag{32}
$$

The first equality in equation (32) follows the definition of $p^t_{\text{RHC}}(x;\theta)$ and the second line follows the definition of the recoverable policy π_{re}. Then, we use lemma B.5 to factor out $\frac{1}{T}$ in the third line. The next inequality follows from the generalized log sum inequality stated in lemma B.4, and we have the first and third terms reduce to 0 and are left with the second term in the next line. We apply lemma B.5 again to the integral using the constant factor $\frac{1}{K+1}$. In addition, to make the equality hold, we multiply the inverse of the constant factor $K+1$ outside the integral. We observe the integral is now the KL divergence between the expert $p_t(x|x_t;\theta_E)$ and one-step-execution of our policy $\frac{1}{K+1}(p(x_t;\theta) + p(x_{t+1};\theta) + \sum_{i=t+2}^{t+K} p(x_i;\theta_E))$. The final inequality follows from the bound derived in equation (31).

For $t = 0$, we have $p^0_{\text{RHC}}(x;\theta) = \frac{1}{T}(p(x_0;\theta) + \sum_{i=1}^{T-1} p(x_i;\theta_E))$. Since the initial state x_0 for expert and our policy are sampled from the same initial state distribution μ, $p(x_0)$ is independent of θ, i.e. $p(x_0;\theta) = p(x_0;\theta_E)$. Therefore, $p_{\text{RHC}}(x;\theta_E) = p(x_0) + \sum_{i=1}^{T-1} p(x_i;\theta_E) = p^0_{\text{RHC}}(x;\theta)$. Moreover, the final global state marginal distribution $p_{\text{RHC}}(x;\theta)$ is the same as the $p^{T-1}_{\text{RHC}}(x;\theta)$,

i.e. $p_{\text{RHC}}(x;\theta) = \sum_{i=0}^{T-1} p(x_i;\theta) = p_{\text{RHC}}^{T-1}(x;\theta)$. For any $t = 1, 2, .., T-1$, we have proved $D_{\text{KL}}(p_{\text{RHC}}^{t-1}(x;\theta) \parallel p_{\text{RHC}}^{t}(x;\theta)) \leq \frac{K+1}{T}\epsilon$.

□

Finally, we are prepared to prove theorem 3.1.

Proof. We evaluate the TV distance over the state marginal distribution between the expert policy and our control law.

$$\begin{aligned} D_{\text{TV}}(p_{\text{RHC}}(x;\theta_E) \parallel p_{\text{RHC}}(x;\theta)) &= D_{\text{TV}}(p_{\text{RHC}}^{0}(x;\theta) \parallel p_{\text{RHC}}^{T-1}(x;\theta)) \\ &\leq \sum_{t=1}^{T-1} D_{\text{TV}}(p_{\text{RHC}}^{t-1}(x;\theta) \parallel p_{\text{RHC}}^{t}(x;\theta)) \end{aligned} \tag{33}$$

The first equality in equation (33) follows from the fact that $p_{\text{RHC}}(x;\theta_E) = p_{\text{RHC}}^{0}(x;\theta)$ and $p_{\text{RHC}}(x;\theta) = p_{\text{RHC}}^{T-1}(x;\theta)$. We use triangle inequality of the TV distance measures to obtain the inequality in the second line.

Recall that by Pinsker's inequality, the total variation (TV) distance is related to Kullback–Leibler (KL) divergence by the following inequality: $D_{\text{TV}}(P \parallel Q) \leq \sqrt{\frac{1}{2} D_{\text{KL}}(P \parallel Q)}$. We apply Pinsker's inequality to each of the TV terms in the second line of equation (33) to bound them by a summation of KL-divergence as shown in the first line of equation (34). Next, given the control sequence distribution for every time step is bounded by ϵ, we apply lemma B.2 to show that the resulting state distribution from the optimal control sequences for each time step t is also bounded by ϵ. Next, we use this result and apply Lemma B.3 to bound the KL-divergence over the global state marginal distribution between two consecutive time steps by $\frac{K+1}{T}\epsilon$. Second line in equation (34) follows from this result and finally we derive the final bound linear in T.

$$\begin{aligned} D_{\text{TV}}(p_{\text{RHC}}(x;\theta_E) \parallel p_{\text{RHC}}(x;\theta)) &\leq \sum_{t=1}^{T-1} \sqrt{\frac{1}{2} D_{\text{KL}}(p_{\text{RHC}}^{t-1}(x;\theta) \parallel p_{\text{RHC}}^{t}(x;\theta))} \\ &\leq \sum_{t=1}^{T-1} \sqrt{\frac{(K+1)\epsilon}{2T}} \\ &= (T-1)\sqrt{\frac{K+1}{T}}\sqrt{\epsilon/2} \\ &\leq T\sqrt{\epsilon/2} \end{aligned} \tag{34}$$

The last line follows from the fact that $K << T$, so $\sqrt{\frac{K+1}{T}} < 1$. □

C Extension to Stochastic Dynamics

RHIRL optimizes the trajectories in the space of control sequences $p(V)$, whereas $V = \{u_0, u_1, u_2, ..., u_{K-1}\}$ is a sequence of controls. If the system is deterministic, we can apply V to the dynamical system $x_{t+1} = f(x_t, v_t)$ with start state x_0 and obtain a state sequence $\tau = (x_0, x_1, \ldots, x_K)$. We recall that total state trajectory cost of V defined in equation (3) as follows:

$$S(V, x_0; \theta) = \sum_{k=0}^{K} g(x_k; \theta)$$

We use the information-theoretic MPC (MPPI) [37] to solve for an optimal control sequence distribution at the current start state x_t in iteration t. The main result of MPPI suggests that, under a deterministic system, the optimal control sequence distribution Q minimizes the "free energy" of the dynamical system and this free energy can be calculated from the cost of the state trajectory under Q. Mathematically, the probability density $p_Q(V^*)$, as shown in equation (4) can be expressed as a

function of the state cost $S(V, x_t; \theta)$, with respect to a Gaussian "base" distribution $B(V_B, \Sigma)$ that depends on the control cost:

$$p_Q(V^*|U_B, \Sigma, x_t; \theta) = \frac{1}{Z} p_B(V^*|U_B, \Sigma) \exp(-\frac{1}{\lambda} S(V^*, x_t; \theta)),$$

where Z is the partition function. Intuitively, this result shows that the control sequence V that results in lower state-trajectory cost $S(V)$ are exponentially more likely to be chosen.

In this section, we extend RHIRL to stochastic dynamics where $x_{t+1} = f(x_t, v_t, \omega_t)$, where ω_t is a random variable that models the independent system noise. More specifically, we assume that $x_{t+1} \sim p(x_{t+1}|x_t, v_t)$. Due to the stochasticity of the dynamics, the state trajectory cost in equation (3) and the optimal control distribution in equation (4) are affected. Therefore, we first redefine the trajectory state cost under stochastic dynamics, then derive the counterpart of equation (4) for the optimal control sequence distribution under the stochastic dynamics, finally we adapt our existing RHIRL algorithm to stochastic dynamics.

C.1 State Trajectory Cost

Due to the stochasticity of the dynamics, given the initial state x_0, we no longer have a one-to-one mapping from the control sequence V to the resulting state trajectory $\tau = (x_0, x_1, ..., x_K)$. Instead, we have a distribution of state trajectories:

$$p(\tau|x_0, V) = \prod_{t=0}^{K-1} p(x_{t+1}|x_t, v_t) \tag{35}$$

To accommodate this change, the trajectory state cost of a control sequence $\tilde{S}(V, x_0; \theta)$ is defined over the distribution of the resulting state trajectories, instead of single trajectory:

$$\tilde{S}(V, x_0; \theta) = \int p(\tau|x_0, V) S(\tau|x_0; \theta) d\tau \tag{36}$$

$$= \int \prod_{t=0}^{K-1} p(x_{t+1}|x_t, v_t) \sum_{t=0}^{K} g(x_t; \theta) d\tau \tag{37}$$

We always measure the preferences over the control sequence V by their resulting state trajectories τ. Hence, when the resulting trajectories changes from a single deterministic sequence of states to a distribution of state trajectories, we adapt our measure of the resulting cost: under the deterministic dynamics where each V uniquely maps to the same state trajectory τ, the state trajectory cost is the cost of that specific trajectory; while under the stochastic dynamics where the same control sequence V maps to a distribution of τ, the state trajectory cost of a control sequence is now defined as the expected state cost of the distribution of trajectory. We measure the state trajectory cost of a control sequence $S(V, x_0; \theta)$, instead of simply a state trajectory cost on the states itself $S(\tau, x_0; \theta)$, because we want to use this measure to directly optimize the control sequence.

C.2 Optimal Control Sequence Distribution

Next, we derive the optimal control sequence distribution under the stochastic dynamics. Our derivation is based on MPPI [37], which uses the "free-energy" principle to derive the optimal control sequence distribution under deterministic dynamics.

Definition 2 (Free-energy ([33], Definition 1)**.** Let $\mathbb{P} \in \mathcal{P}(\mathcal{Z})$ and the function $\mathcal{J}(x)$: $\mathcal{Z} \to \mathbb{R}$ be a measurable function. The the term:

$$\mathbb{E}(\mathcal{J}(x)) = \log \int \exp(\rho \mathcal{J}(x)) d\mathbb{P} \tag{38}$$

is called free energy of $\mathcal{J}(x)$ with respect to $\mathbb{P}$, ρ is a constant.

Now we have the free-energy of a control system under stochastic dynamics as stated below. It has a "Gaussian" base control sequence distribution $B(U_B, \Sigma)$ such that its control sequence distribution

follows $p_B(V^*|U_B, \Sigma)$ whereas Σ is the Gaussian control noise covariance matrix. $\tilde{S}(V;\theta)$ denotes the state trajectory cost function.

$$\mathcal{F}(\tilde{S}, p_B, x_0, \lambda; \theta) = \log(\mathbb{E}_{p_B}[\exp(-\frac{1}{\lambda}\tilde{S}(V, x_0; \theta))]), \tag{39}$$

$\lambda \in \mathbb{R}^+$ is the inverse temperature of the control system.

Suppose now we have another control sequence distribution with probability measure $p(V)$ and these two distributions are absolutely continuous, then we can rewrite the free-energy w.r.t $p_B(V)$ using the expectation over the density of $p(V)$ use the standard importance sampling trick:

$$\begin{aligned}
\mathcal{F}(\tilde{S}, p_B, x_0, \lambda; \theta) &= \log(\mathbb{E}_{p_B}[\exp(-\frac{1}{\lambda}\tilde{S}(V, x_0; \theta))]) && (40)\\
&= \log(\mathbb{E}_{p_B}[\exp(-\frac{1}{\lambda}\tilde{S}(V, x_0; \theta)\frac{p_B(V^*|U_B, \Sigma)}{p(V)})]) && (41)\\
&\geq \mathbb{E}_p[\log(\exp(-\frac{1}{\lambda}\tilde{S}(V, x_0; \theta)\frac{p_B(V^*|U_B, \Sigma)}{p(V)}))] && (42)\\
&= -\frac{1}{\lambda}\mathbb{E}_p[\tilde{S}(V, x_0; \theta) + \lambda\log(\frac{p(V)}{p_B(V^*|U_B, \Sigma)})] && (43)\\
&= -\frac{1}{\lambda}(\mathbb{E}_p[\tilde{S}(V, x_0; \theta)] + \lambda\mathbb{E}_p[\log(\frac{p(V)}{p_B(V^*|U_B, \Sigma)})]) && (44)\\
&= -\frac{1}{\lambda}(\mathbb{E}_p[\tilde{S}(V, x_0; \theta)] + \lambda D_{\text{KL}}(p(V)||p_B(V^*|U_B, \Sigma))) && (45)
\end{aligned}$$

Therefore, we have

$$\mathcal{F}(S, p_B, x_0, \lambda; \theta) \geq -\frac{1}{\lambda}(\mathbb{E}_p[\tilde{S}(V, x_0; \theta)] + \lambda D_{\text{KL}}(p(V)||p_B(V^*|U_B, \Sigma))) \tag{46}$$

The right-hand side is the lower bound of the free-energy of the control system. We use $p_Q(V)$ to denote the optimal control sequence distribution. This distribution is only optimal if and only if the bound in the equation above is tight, i.e. $\mathcal{F}(\tilde{S}, p_B, x_0, \lambda; \theta) = -\frac{1}{\lambda}\mathbb{E}_{p_Q}[\tilde{S}(V, x_0; \theta)] + \lambda D_{\text{KL}}(p_Q(V)||p_B(V^*|U_B, \Sigma))$.

We claim that the optimal control sequence distribution $\tilde{p_Q}$ under the stochastic dynamics is as follows:

$$\tilde{p_Q}(V^*|U_B, \Sigma, x_0; \theta) = \frac{1}{Z}\exp(-\frac{1}{\lambda}\tilde{S}(V, x_0; \theta))p_B(V^*|U_B, \Sigma), \tag{47}$$

whereas $Z = \int \exp(-\frac{1}{\lambda}\tilde{S}(V, x_0; \theta))p_B(V^*|U_B, \Sigma)dV$ is the partition function.

We prove that equation (47) is the optimal control sequence distribution under stochastic dynamics by showing that this $\tilde{p_Q}(V^*|U_B, \Sigma, x_0; \theta)$ tightens the bound of free-energy in equation (46). We substitute $\tilde{p_Q}(V^*|U_B, \Sigma, x_0; \theta)$ into the RHS of equation (46) and simplify the expression of the KL-divergence:

$$\begin{aligned}
\mathcal{F}(\tilde{S}, p_B, x_0, \lambda; \theta) &\geq -\frac{1}{\lambda}(\mathbb{E}_{\tilde{p_Q}}[\tilde{S}(V, x_0; \theta)] + \lambda D_{\text{KL}}(\tilde{p_Q}(V^*|U_B, \Sigma, x_0; \theta)||p_B(V^*|U_B, \Sigma))) && (48)\\
&= -\frac{1}{\lambda}(\mathbb{E}_{\tilde{p_Q}}[\tilde{S}(V, x_0; \theta)] + \lambda\mathbb{E}_{\tilde{p_Q}}[\log(\frac{\tilde{p_Q}(V^*|U_B, \Sigma, x_0; \theta)}{p_B(V^*|U_B, \Sigma)})]) && (49)\\
&= -\frac{1}{\lambda}\mathbb{E}_{\tilde{p_Q}}[\tilde{S}(V, x_0; \theta)] - \mathbb{E}_{\tilde{p_Q}}[\log(\frac{\frac{1}{Z}\exp(-\frac{1}{\lambda}\tilde{S}(V, x_0; \theta))p_B(V^*|U_B, \Sigma)}{p_B(V^*|U_B, \Sigma)})] && (50)\\
&= -\frac{1}{\lambda}\mathbb{E}_{\tilde{p_Q}}[\tilde{S}(V, x_0; \theta)] - (\frac{1}{\lambda}\mathbb{E}_{\tilde{p_Q}}[\tilde{S}(V, x_0; \theta)] - \log(Z)) && (51)
\end{aligned}$$

Next we substitute the expression for the partition function Z and we found that the RHS is exactly the definition of the the free-energy of the control system with base distribution $B(U_B, \Sigma)$:

$$\mathcal{F}(\tilde{S}, p_B, x_0, \lambda; \theta) \geq \log(Z) \tag{52}$$

$$= \log(\int \exp(-\frac{1}{\lambda}\tilde{S}(V, x_0; \theta))p_B(V^*|U_B, \Sigma)dV) \tag{53}$$

$$= \log(\mathbb{E}_{p_B}[\exp(-\frac{1}{\lambda}\tilde{S}(V, x_0; \theta))]) \tag{54}$$

$$= \mathcal{F}(\tilde{S}, p_B, x_0, \lambda; \theta) \tag{55}$$

The final equality forces the inequality to be tight. Therefore, $p_Q(V^*|U_B, \Sigma, x_0; \theta)$ in equation (47) is the optimal control sequence distribution under the stochastic dynamics.

We observe that the optimal control sequence distribution under deterministic system $p_Q(V^*|U_B, \Sigma, x_0; \theta)$ in equation (4) and that under the stochastic dynamics $\tilde{p_Q}(V^*|U_B, \Sigma, x_0; \theta)$ in equation (47) only differs in the calculation of the state trajectory cost of the control sequences. Intuitively, it means that under deterministic dynamics, we choose the control sequence V that will, for sure, leads to a state trajectory with lower cost; while when we extend to stochastic dynamics, the control sequence V that results in lower state-trajectory cost $S(V)$ in expectation are exponentially more likely to be chosen. Practically, now we need more samples for a single control sequence to compute the expectation in equation (37).

C.3 RHIRL under Stochastic Dynamics

Next, we adapt our RHIRL algorithm to this new state trajectory cost measure $\tilde{S}(V, x_0; \theta)$ and the optimal control sequences $\tilde{p_Q}(V^*|U_B, \Sigma, x_0; \theta)$ under stochastic dynamics. We recall that under the deterministic dynamics, RHIRL uses importance sampling in equation (12) to estimate the $\frac{\partial \mathcal{L}}{\partial \theta}$ so as to update the cost function parameter θ:

$$\frac{\partial}{\partial \theta}\mathcal{L}(\theta; D, x_0) \approx \frac{1}{N}\sum_{i=1}^{N}\frac{1}{\lambda}\frac{\partial}{\partial \theta}S(V_i, x_0; \theta) - \frac{1}{M}\sum_{j=1}^{M}\frac{1}{\lambda}w(V_j)\frac{\partial}{\partial \theta}S(V_j, x_0; \theta),$$

whereas the N control sequences in the first term are from the expert demonstration D_t and the M control sequences in the second term are from our approximated optimal control sequence distribution $p_Q(V^*)$, and $w(V_j)$ is the importance sampling weight.

To estimate $\frac{\partial \mathcal{L}}{\partial \theta}$, we need to calculate/approximate the importance sampling weight $w(V)$, and the derivative of the state trajectory cost $\frac{\partial}{\partial \theta}S(V, x_0; \theta)$ w.r.t θ. We recall that the importance sampling weight $w(V)$ depends on the state trajectory cost $S(V, x_0; \theta)$ in equation (11):

$$w(V) \propto \exp(-\frac{1}{\lambda}\left(S(V, x_0; \theta) + \lambda\sum_{k=0}^{K-1} u_k^\intercal \Sigma^{-1} v_k)\right)$$

Under the deterministic dynamics, the importance sampling weight $w(V)$ is estimated using Monte-Carlo approximation with M state trajectory samples as follows:

$$w(V) \approx \frac{\exp\left(-\frac{1}{\lambda}(S(V, x_0; \theta) + \lambda\sum_{k=0}^{K-1} u_k^\intercal \Sigma^{-1} v_k)\right)}{\sum_{j=0}^{M-1}\exp\left(-\frac{1}{\lambda}(S(V_j, x_0; \theta) + \lambda\sum_{k=0}^{K-1} u_k^{j\intercal} \Sigma^{-1} v_k^j)\right)} \tag{56}$$

$$= \frac{\exp\left(-\frac{1}{\lambda}(\sum_{t=0}^{K} g(x_t; \theta) + \lambda\sum_{k=0}^{K-1} u_k^\intercal \Sigma^{-1} v_k)\right)}{\sum_{j=0}^{M-1}\exp\left(-\frac{1}{\lambda}(\sum_{t=0}^{K} g(x_t^j; \theta) + \lambda\sum_{k=0}^{K-1} u_k^{j\intercal} \Sigma^{-1} v_k^j)\right)} \tag{57}$$

Since the state trajectory cost $S(V, x_0; \theta)$ is a linear sum of the cost of all states, $\frac{\partial S}{\partial \theta}$ can be directly computed as follows:

$$\frac{\partial}{\partial \theta}S(V, x_0; \theta) = \frac{\partial}{\partial \theta}\sum_{t=0}^{K} g(x_t; \theta) = \sum_{t=0}^{K}\frac{\partial}{\partial \theta}g(x_t; \theta) \tag{58}$$

To extend RHIRL to stochastic dynamics, when the state trajectory cost function is now $\tilde{S}(V, x_0; \theta)$, we need to redefine how to estimate $\tilde{w}(V)$ and consequently $\frac{\partial \tilde{S}}{\partial \theta}$.

When extend to stochastic dynamic, we have the following:

$$\tilde{w}(V) \propto \exp(-\frac{1}{\lambda}\left(\tilde{S}(V, x_0; \theta) + \lambda \sum_{k=0}^{K-1} u_k^\intercal \Sigma^{-1} v_k)\right) \tag{59}$$

and we still adopts Monte-carlo sampling to approximate $\tilde{w}(V)$. However, since $\tilde{S}(V, x_0; \theta)$ now measures the expected state trajectory cost under the stochastic dynamics, we need to go one step further and use sampling to estimate $\tilde{S}(V, x_0; \theta)$ using M^s number of state trajectories $\tau_h = (x_0, x_1^h, ..., x_K^h)$ per (x_0, V) pair:

$$\tilde{S}(V, x_0; \theta) \approx \frac{1}{M^s} \sum_{h=0}^{M^s-1} \sum_{t=0}^{K} g(x_t^h; \theta) \tag{60}$$

Therefore, now the importance sampling weight $\tilde{w}(V)$ is approximated from $M \times M^s$ state trajectories, with M^s trajectories from each (x_0, V_j) pair as follows:

$$\tilde{w}(V) \approx \frac{\exp\left(-\frac{1}{\lambda}(\tilde{S}(V, x_0; \theta) + \lambda \sum_{k=0}^{K-1} u_k^\intercal \Sigma^{-1} v_k)\right)}{\sum_{j=0}^{M-1} \exp\left(-\frac{1}{\lambda}(\tilde{S}(V_j, x_0; \theta) + \lambda \sum_{k=0}^{K-1} u_k^{j\intercal} \Sigma^{-1} v_k^j)\right)} \tag{61}$$

$$\approx \frac{\exp\left(-\frac{1}{\lambda}(\frac{1}{M^s} \sum_{h=0}^{M^s-1} \sum_{t=0}^{K} g(x_t^h; \theta) + \lambda \sum_{k=0}^{K-1} u_k^\intercal \Sigma^{-1} v_k)\right)}{\sum_{j=0}^{M-1} \exp\left(-\frac{1}{\lambda}(\frac{1}{M^s} \sum_{h=0}^{M^s-1} \sum_{t=0}^{K} g(x_t^{j^h}; \theta) + \lambda \sum_{k=0}^{K-1} u_k^{j\intercal} \Sigma^{-1} v_k^j)\right)} \tag{62}$$

We emphasize that under the stochastic dynamics, we use the state trajectory samples to estimate both the state trajectory cost $\tilde{S}(V, x_0; \theta)$ and the importance sampling weight $\tilde{w}(V)$. Since the $\tilde{S}$ now measures the expected cost over a distribution of trajectories, we need more samples to estimate $\tilde{w}(V)$ compared to the deterministic setting.

Moreover, in the final $\frac{\partial \mathcal{L}}{\partial \theta}$, we need to differentiate $\tilde{S}(V, x_0; \theta)$ w.r.t. θ. Since $\tilde{S}$ is estimated from sampling, we have:

$$\frac{\partial}{\partial \theta} \tilde{S}(V, x_0; \theta) \approx \frac{\partial}{\partial \theta} \frac{1}{M^s} \sum_{h=0}^{M^s-1} \sum_{t=0}^{K} g(x_t^h; \theta) \approx \frac{1}{M^s} \sum_{h=0}^{M^s-1} \sum_{t=0}^{K} \frac{\partial}{\partial \theta} g(x_t^h; \theta) \tag{63}$$

Finally, we summarize how to extend RHIRL to stochastic dynamics. In stochastic dynamics, each control sequence V will map to a distribution of state trajectory $p(\tau|V, x_0)$. Hence, we adapt our measure of state trajectory cost $S(V, x_0; \theta)$ from a single trajectory to be the expected cost over a distribution of state trajectories $\tilde{S}(V, x_0; \theta)$ in equation (37). Next, we revise the optimal control sequence distribution to a stochastic setting $\tilde{p_Q}(V^*|U_B, \Sigma, x_0; \theta)$ in equation (47). More specifically, we show under the stochastic dynamics, the optimal control sequences V^* is chosen based on the expected cost of its resulting state trajectories. Finally, under this new state trajectory cost $\tilde{S}(V, x_0; \theta)$ and the optimal control sequence distribution $\tilde{p_Q}(V^*|U_B, \Sigma, x_0; \theta)$, we adapt the approximation of the importance sampling weight $\tilde{w}(V)$ and consequently the gradient of the overall loss w.r.t θ by adding one more sampling process to estimate the new state trajectory cost. Moreover, in practice, we can use the same set of samples to estimate both state trajectory cost $\tilde{S}(V, x_0; \theta)$ and the importance sampling weights $\tilde{w}(V)$.

D Experimental Details

In this section, we list down the implementation details of RHIRL and the baselines. The code is included in the supplementary material. We also report the hyperparameters used in the experiments, the detailed network architectures, training procedures and evaluation procedures used for our experiments.

D.1 Practical Issues of RHIRL

Control Noise Covariance Approximation

The actual control noise covariance Σ is unknown to RHIRL and the baselines. However, RHIRL uses the noise covariance matrix Σ to sample the controls around the nominal control (Algorithm 1, line 6) and calculate the quadratic control cost in Equation (37). Since we have no access to the true Σ, RHIRL approximates Σ as a constant factor of the identity matrix βI, whereas β is the hyperparameter we optimize using grid search and I is the identify matrix with its width equals to the dimension of the action space. Therefore, instead of sampling the controls from $\mathcal{N}(V|U, \Sigma)$, we sample from $\mathcal{N}(V|U, \beta I)$ in practice. We also use βI in Equation (37) to replace the unknown Σ. Even in the noise-free environment, we set β to a non-zero value to foster exploration; otherwise, the importance sampling degenerates to the single nominal control.

Our experiment shows that RHIRL is robust to the choice of β: the cost learning performs well even if $\beta I \neq \Sigma$. This may be attributed to the fact that we jointly optimize the state cost function and β. Therefore the learned state cost function may compensate for the inaccurate approximation for Σ.

Numerical Stability

Equation(11) forms the basis of importance sampling and estimation. However, the learned cost can be a huge negative number, which causes numerical instability in estimating the importance weights. To mitigate this issue, we subtract the minimum trajectory cost S_{min} from all rollouts to improve the numeric stability. Since subtracting the same number from all rollouts does not change the order of the preference, this operation does not affect the optimality of our derivation.

Nominal Control Initialization and Local Optimality

RHIRL samples around the nominal control sequence to collect the samples for importance sampling. However, if the initial nominal control sequence performs poorly, it is not easy to generate any good samples to improve the current control sequences. To mitigate this problem, we add an exploration strategy to the sampling process in (Algorithm 1, line 6): with probability α, we continue with the standard sampling strategy to sample around the nominal control; with probability $1 - \alpha$ we sample uniformly from the entire action space. This helps RHIRL to correct from the unsuitable nominal control initialization and also helps RHIRL to escape the local optimal solution. We set $\alpha = 0.5$ for all tasks in our experiments.

Control Smoothness

Updating the optimal control by importance sampling might cause some jerk in the control space. In order to make the control change smoothly in its local space, we apply a Savitzky–Golay filter over the time horizon dimension to constrain the control that does not change too much over the time horizon.

D.2 Training

We list the hyper-parameters of RHIRL for different tasks. These hyper-parameters were selected via grid search.

Task	K	β	batch size	λ	lr	weight decay
Pendulum-v0	20	0.8	50	0.10	1e-4	8e-5
LunarLanderContinuous-v2	40	0.6	200	0.10	1e-4	8e-5
Hopper-v2	20	0.8	100	0.10	1e-4	8e-5
Walker2d-v2.	30	0.6	150	0.10	1e-4	8e-5
Ant-v2	15	1.2	200	0.10	1e-4	8e-5
CarRacing-v0	15	1.0	200	0.10	1e-4	8e-5

The implementation of the baselines (f-IRL, AIRL and GAIL) are adapted from f-IRL's [27] official repository. We use the hyperparameters reported in f-IRL for the MuJoCo tasks and performed grid search on the hyperparameters for the rest of the tasks. SAC[12] is used as the base MaxEnt RL algorithm for both expert policy and the baselines optimization algorithm. We use a tanh squashed Gaussian as the policy network for Pendulum-v0, LunarLander-v2, and the MuJoCo tasks; and we

use a Gaussian Convolutional policy as the policy network for CarRacing-v0. The mean and std of the Gaussian are parameterized by a ReLU MLP of size (64, 64). Adam is used as the optimizer. We use the reported SAC temperature,$\alpha = 0.2$, reward scale $c = 0.2$, and gradient penalty coefficient $\lambda = 4.0$. The rest of the hyperparameters for f-IRL, GAIL and AIRL are listed below.

Task	SAC learning rate	SAC replay buffer size	Reward/Value model learning rate	l2 weight decay
Pendulum-v0	1e-4	100000	1e-5	1e-3
LunarlanderContinuous-v2	1e-3	100000	1e-5	1e-3
Hopper-v2	1e-5	1000000	1e-5	1e-3
Walker2d-v2	1e-5	1000000	1e-5	1e-3
Ant-v2	3e-4	1000000	1e-4	1e-3
CarRacing-v0	4e-4	10000000	1e-4	1e-3

D.3 Reward Function and Discriminator Network Architectures

We use the same neural network architecture to parameterize the cost-function/reward-function/discriminator for all methods. For continuous control task with raw state input, i.e. pendulum, lunarlander and the MuJoCo tasks, we use two-layer of MLP with ReLU activation function to parameterized the cost function/discriminator. The hidden size for Pendulum-v0 is (32, 32), and (64, 64) for the rest of the tasks.

For continuous control task with image input, i.e. carracing, we use a four convolutional layer with kernel size 3×3 as the feature extractor. The output of the CNN layer is vector with size (128,) and is fed into the same reward network as describe above.

D.4 Additional Experiments Results

We report the average returns and the standard deviation for Table 1 and Table 2 in Table 3 and Table 4 respectively. The mean and standard deviation computed from 3 trials for each entry of the tables.

Table 3: Performance of RHIRL, f-IRL, GAIL, and AIRL. We report the mean and the standard deviation of the policy returns using the ground-truth task reward. Higher values indicate better performance.

		No Noise $\Sigma = 0$	**Mild Noise** $\Sigma = 0.2$	**High Noise** $\Sigma = 0.5$
Pendulum	Expert	-154.69 ± 50.05	-156.50 ± 70.72	-168.54 ± 80.89
	RHIRL	-125.95 ± 1.21	-122.33 ± 3.44	-132.39 ± 10.36
	f-IRL	-121.94 ± 97.21	-127.51 ± 104.55	-197.36 ± 106.92
	AIRL	-131.64 ± 1.16	-184.62 ± 88.16	-203.12 ± 80.57
	GAIL	-207.05 ± 57.41	-207.14 ± 57.52	-253.85 ± 181.84
LunarLander	Expert	235.13 ± 43.59	222.65 ± 56.35	164.52 ± 36.79
	RHIRL	246.39 ± 10.96	233.73 ± 23.75	198.23 ± 47.8
	f-IRL	179.03 ± 9.19	141.73 ± 11.81	121.67 ± 22.77
	AIRL	174.49 ± 35.17	132.76 ± 85.59	95.61 ± 19.25
	GAIL	169.98 ± 15.43	125.5 ± 16.78	100.24 ± 79.04
Hopper	Expert	3222.48 ± 390.65	3159.32 ± 520.00	2887.72 ± 483.93
	RHIRL	3071.63 ± 122.03	3121.72 ± 278.98	2776.2 ± 345.90
	f-IRL	3080.34 ± 458.96	2580.19 ± 637.21	1270.24 ± 539.84
	AIRL	18.9 ± 0.79	33.52 ± 3.86	18.38 ± 7.84
	GAIL	2642.59 ± 187.33	1576.25 ± 1051.98	702.33 ± 151.37
Walker2d	Expert	4999.47 ± 55.99	4500.43 ± 114.48	3624.48 ± 95.05
	RHIRL	4939.44 ± 100.28	4473.332 ± 324.34	3446.55 ± 507.89
	f-IRL	4927.92 ± 529.95	3697.36 ± 711.56	2831.91 ± 993.76
	AIRL	-2.51 ± 0.69	22.24 ± 10.74	6.5 ± 5.03
	GAIL	2489.04 ± 813.31	2884.35 ± 59.88	1840.62 ± 778.3
Ant	Expert	5759.22 ± 173.57	2557.37 ± 501.95	252.62 ± 91.44
	RHIRL	4987.67 ± 149.2	2373.32 ± 529.3	230.8 ± 253.39
	f-IRL	5022.42 ± 108.07	2034.87 ± 262.29	197.2 ± 200.45
	AIRL	1000.4 ± 0.79	849.05 ± 30.15	-7.43 ± 6.01
	GAIL	2784.87 ± 301.66	1022.04 ± 580.49	-416.69 ± 292.23
CarRacing	Expert	903.25 ± 0.23	702.01 ± 0.3	281.12 ± 0.34
	RHIRL	359.61 ± 40.32	206.21 ± 19.87	53.97 ± 3.24
	f-IRL	85.45 ± 47.4	18.32 ± 27.89	2.04 ± 13.8
	AIRL	-21.97 ± 2.67	-25.25 ± 5.98	-32.31 ± 7.43
	GAIL	2.62 ± 3.41	-7.65 ± 4.77	-15.88 ± 5.89

Table 4: Generalization of learned cost functions over different noise levels.

		Noise-free for learning	**Noise Level Σ for Testing** 0.2	0.5
Pendulum	RHIRL	-125.95 ± 1.21	-125.01 ± 4.53	-126.4 ± 7.73
	f-IRL	-121.94 ± 97.21	-199.44 ± 96.99	-220.74±79.75
	AIRL	-131.64 ± 1.16	-247.86±11.44	-304.48±20.78
	GAIL	-207.05 ± 57.41	-220.6± 69.82	-270.81± 79.68
LunarLander	RHIRL	246.39 ± 10.96	205.66±24.67	175.82±52.12
	f-IRL	179.03 ± 9.19	121.80±20.94	102.06±22.31
	AIRL	174.49 ± 35.17	31.46±9.68	22.29±14.01
	GAIL	169.98 ± 15.43	101.80 ± 23.12	78.33± 24.15
Hopper	RHIRL	3071.63 ± 122.03	2577.28 ±409.33	2152.08 ±342.21
	f-IRL	3080.34 ± 458.96	2110.52±26.71	1984.29±31.88
	AIRL	18.9 ± 0.79	18.86±4.80	8.78±10.89
	GAIL	2642.59 ± 187.33	215.29± 27.76	132.15± 30.20
Walker2d	RHIRL	4939.44 ± 100.28	4039.44±39.2	3440.23±531.08
	f-IRL	4927.92 ± 529.95	2976.66±396.57	1090.11±1389.56
	AIRL	-2.51 ± 0.69	1380.84±364.95	1787.15±230.94
	GAIL	2489.04 ± 813.31	103.15± 121.84	124.15±82.84
Ant	RHIRL	4987.67 ± 149.2	3192.82±162.12	867.08±204.28
	f-IRL	5022.42 ± 108.07	2042.41±129.89	472.77±110.2
	AIRL	1000.4 ± 0.79	845.69 ±29.01	0.69 ±20.49
	GAIL	2784.87 ± 301.66	-6.41± 21.17	-79.89± 142.43
CarRacing	RHIRL	359.61 ± 40.32	261.78±54.44	110.12±58.90
	f-IRL	85.45 ± 47.4	16.12±67.82	-24.78±2.12
	AIRL	-21.97 ± 2.67	-27.09±6.65	-23.96±4.11
	GAIL	2.62 ± 3.41	-6.41±3.22	-49.89±7.98