MEREQ: Max-Ent Residual-Q Inverse RL for Sample-Efficient Alignment from Intervention

Yuxin Chen[*,1], **Chen Tang**[*,2], **Jianglan Wei**[1], **Chenran Li**[1], **Ran Tian**[1], **Xiang Zhang**[1], **Wei Zhan**[1], **Peter Stone**[2,3], **Masayoshi Tomizuka**[1]
[*]Denotes equal contribution
[1]*University of California, Berkeley* [2]*The University of Texas at Austin* [3]*Sony AI*

Abstract: Aligning robot behavior with human preferences is crucial for deploying embodied AI agents in human-centered environments. A promising solution is interactive imitation learning from human intervention, where a human expert observes the policy's execution and provides interventions as feedback. However, existing methods often fail to utilize the prior policy efficiently to facilitate learning, thus hindering sample efficiency. In this work, we introduce Maximum-Entropy Residual-Q Inverse Reinforcement Learning (MEREQ)[1], designed for sample-efficient alignment from human intervention. Instead of inferring the complete human behavior characteristics, MEREQ infers a *residual reward function* that captures the discrepancy between the human expert's and the prior policy's underlying reward functions. Residual Q-Learning (RQL) is then employed to align the policy with human preferences using the inferred reward function. Extensive evaluations on simulated and real-world tasks show that MEREQ achieves sample-efficient alignment from human intervention compared to baselines.

Keywords: Interactive imitation learning, Learning from human feedback, Inverse reinforcement learning

1 Introduction

Recent progress in embodied AI has enabled advanced robots to perform complex real-world tasks that go beyond pre-scripted routines and highly controlled environments. Increasing research attention has been focused on how to align their behavior with human preferences [1, 2], which is crucial for their deployment in human-centered environments. One promising approach is interactive imitation learning, where a pre-trained policy can interact with a human and align its behavior to the human's preference through human feedback [2, 3]. In this work, we focus on interactive imitation learning using *human interventions* as feedback. In this setting, the human expert observes the policy during task execution and intervenes whenever it deviates from their preferred behavior. A straightforward approach [4, 5, 6] is to update the policy through behavior cloning (BC) [7]—maximizing the likelihood of the collected intervention samples under the learned policy distribution. However, BC ignores the sequential nature of decision-making, leading to compounded errors [8]. Additionally, Jiang et al. [9] pointed out that these approaches are not ideal for the fine-tuning setting, since the policies are fine-tuned to fit solely the collected intervention data, thus suffering from catastrophic forgetting, which hinders sample efficiency.

We instead study the learning-from-intervention problem within the inverse reinforcement learning (IRL) framework [10, 11]. IRL models the expert as a sequential decision-making agent who maximizes cumulative returns based on their internal reward function, and infers this reward function from expert demonstrations. IRL inherently accounts for the sequential nature of human decision-making and the effects of transition dynamics [12]. Maximum-entropy IRL (MaxEnt-IRL) further

[1]Website: `https://thomaschen98.github.io/mereq.github.io/`.

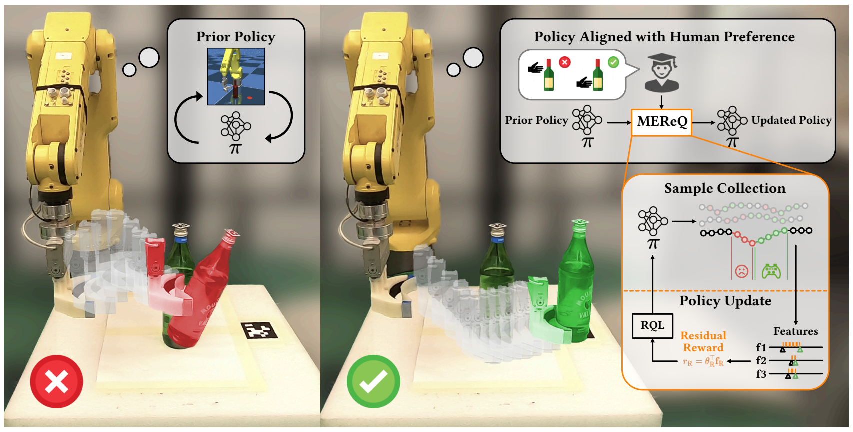


Figure 1: **Overview of MEREQ**, designed for sample-efficient alignment from human intervention. From human intervention samples, MEREQ infers a residual reward that captures the discrepancy between the human expert's and the prior policy's underlying reward functions via maximum-entropy inverse reinforcement and then updates the prior policy with Residual Q-Learning (RQL).

accounts for the sub-optimality in human behavior by favoring randomness in the policy that is learned from the inferred reward function [13, 14, 11]. However, directly applying IRL to fine-tune a prior policy from interventions can still be inefficient. The prior policy is ignored in the learning process, except as an initialization for the learning policy. Consequently, like other approaches, it fails to effectively leverage the prior policy to reduce the number of intervention samples needed.

To address this shortcoming, this paper introduces **M**aximum-**E**ntropy **Re**sidual-**Q** Inverse Reinforcement Learning (MEREQ) for *sample-efficient alignment from human intervention*. The key insights behind MEREQ are to infer a *residual reward function* that captures the discrepancy between the human expert's internal reward function and that of the prior policy, rather than inferring the full human reward function from interventions. MEREQ then employs Residual Q-Learning (RQL) [15] to fine-tune and align the policy with the unknown expert reward, leveraging the inferred residual reward function. We show that MEREQ can effectively align a prior policy with fewer interventions than baselines in both simulation and real-world tasks.

2 Related Work

Interactive imitation learning (IL) utilizes human feedback to align policies with human behavior preference [2, 3]. Forms of human feedback include preference [16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26], interventions [6, 27, 28, 29, 30, 4, 31, 32], scaled feedback [33, 34, 35, 36, 37, 38, 39, 40] and rankings [41]. Like ours, several approaches [26, 42, 43, 44, 45] opt to infer the internal reward function of humans from their feedback and update the policy using the inferred reward function. While these methods have demonstrated improved performance and sample efficiency compared to those without a human in the loop [5], further enhancing efficiency beyond the sample collection pattern has not been thoroughly explored. Our method addresses this gap by building on a prior policy and inferring only the residual reward to improve sample efficiency. Based on similar motivation, Jiang et al. introduced TRANSIC [9], which learns a residual policy from human corrections and integrates it with the prior policy for autonomous execution. Their framework addresses the sim-to-real gaps through human corrections, whereas our method targets the problem where there is a mismatch in rewards between the prior policy and the human subject.

3 Preliminaries

In this section, we briefly review two techniques used in MEREQ, which are RQL and MaxEnt-IRL, to establish the foundations for the main technical results.

3.1 Policy Customization and Residual Q-Learning

Li et al. [15] introduced a new problem setting termed *policy customization*. Given a prior policy, the goal is to find a new policy that jointly optimizes 1) the task objective the prior policy is designed for; and 2) additional task objectives specified by a downstream task. The authors proposed RQL as an initial solution. Formally, RQL assumes the prior policy $\pi : \mathcal{S} \times \mathcal{A} \mapsto [0, \infty)$ is a max-ent policy solving a Markov Decision Process (MDP) defined by the tuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}, r, p, \rho_0, \gamma)$, where $\mathcal{S} \in \mathbb{R}^S$ is the state space, $\mathcal{A} \in \mathbb{R}^A$ is the action space, $r : \mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}$ is the reward function, $p : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \mapsto [0, \infty)$ represents the probability density of the next state $\mathbf{s}_{t+1} \in \mathcal{S}$ given the current state $\mathbf{s}_t \in \mathcal{S}$ and action $\mathbf{a}_t \in \mathcal{A}$, ρ_0 is the starting state distribution, and $\gamma \in [0, 1)$ is the discount factor. That is to say, π follows the Boltzmann distribution [46]:

$$\pi(\mathbf{a}|\mathbf{s}) = \frac{1}{Z_{\mathrm{s}}} \exp\left(\frac{1}{\alpha} Q^{\star}(\mathbf{s}, \mathbf{a})\right), \tag{1}$$

where $Q^{\star}(\mathbf{s}, \mathbf{a})$ is the soft Q-function as defined in [46], which satisfies the soft Bellman equation.

Policy customization is then formalized as finding a max-ent policy $\hat{\pi} : \mathcal{S} \times \mathcal{A} \mapsto [0, \infty)$ for a new MDP defined by $\hat{\mathcal{M}} = (\mathcal{S}, \mathcal{A}, r + r_{\mathrm{R}}, p, \rho_0, \gamma)$, where $r_{\mathrm{R}} : \mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}$ is a *residual reward* function that quantifies the discrepancy between the original task objective and the customized task objective for which the policy is being customized. Given π, RQL finds this customized policy without knowledge of the prior reward r. Specifically, define the soft Bellman update operator [46, 47] as:

$$\hat{Q}_{t+1}(\mathbf{s}, \mathbf{a}) = r_{\mathrm{R}}(\mathbf{s}, \mathbf{a}) + r(\mathbf{s}, \mathbf{a}) + \gamma \mathbb{E}_{\mathbf{s}' \sim p(\cdot|\mathbf{s}, \mathbf{a})} \left[\hat{\alpha} \log \int_{\mathcal{A}} \exp\left(\frac{1}{\hat{\alpha}} \hat{Q}_t(\mathbf{s}', \mathbf{a}')\right) d\mathbf{a}' \right], \tag{2}$$

where $\hat{Q}_t$ is the estimated soft Q-function at the t^{th} iteration. RQL introduces a *residual* Q-function defined as $Q_{\mathrm{R},t} := \hat{Q}_t - Q^{\star}$. It was shown that $Q_{\mathrm{R},t}$ can be learned without knowing r and the customized policy can be defined using $Q_{\mathrm{R},t}$ and π.

RQL considers the case where r_{R} is specified. In this work, we aim to customize the policy towards a human behavior preference, under the assumption that r_{R} is unknown a priori. MEREQ is proposed to infer r_{R} from interventions and customize the policy towards the inferred residual reward.

3.2 Maximum-Entropy Inverse Reinforcement Learning

In the IRL setting, an agent is assumed to optimize a reward function defined as a linear combination of a set of *features* $\mathbf{f} : \mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}^f$ with weights $\theta \in \mathbb{R}^f$: $r = \theta^{\top} \mathbf{f}(\zeta)$. Here $\mathbf{f}(\zeta)$ is the trajectory *feature counts*, $\mathbf{f}(\zeta) = \sum_{(\mathbf{s}_i, \mathbf{a}_i)} \mathbf{f}(\mathbf{s}_i, \mathbf{a}_i)$, which are the sum of the state-action features $\mathbf{f}(\mathbf{s}_i, \mathbf{a}_i)$ along the trajectory ζ. IRL [10] aligns the feature expectations between an observed expert and the learned policy. However, multiple reward functions can yield the same optimal policy, and different policies can result in identical feature counts [11]. One way to resolve this ambiguity is by employing the principle of maximum entropy [48], where policies that yield equivalent expected rewards are equally probable, and those with higher rewards are exponentially favored:

$$p(\zeta|\theta) = \frac{p(\zeta)}{Z_{\zeta}(\theta)} \exp\left(\theta^{\top} \mathbf{f}(\zeta)\right) = \frac{p(\zeta)}{Z_{\zeta}(\theta)} \exp\left[\sum_{(\mathbf{s}_i, \mathbf{a}_i)} \theta^{\top} \mathbf{f}(\mathbf{s}_i, \mathbf{a}_i)\right], \tag{3}$$

where $Z_{\zeta}(\theta)$ is the *partition function* defined as $\int p(\zeta) \exp\left(\theta^{\top} \mathbf{f}(\zeta)\right) d\zeta$ and $p(\zeta)$ is the trajectory prior. The optimal weight $\theta^{\star}$ is obtained by maximizing the likelihood of the observed data:

$$\theta^{\star} = \arg\max_{\theta} \mathcal{L} = \arg\max_{\theta} \log p(\tilde{\zeta}|\theta), \tag{4}$$

where $\tilde{\zeta}$ represents the demonstration trajectories. The optima can be obtained using gradient-based optimization with gradient defined as $\nabla_{\theta} \mathcal{L} = \mathbf{f}(\tilde{\zeta}) - \int p(\zeta|\theta) \mathbf{f}(\zeta) d\zeta$. At the maxima, the feature expectations align, ensuring that the learned policy's performance matches the demonstration, regardless of the specific reward weights the agent aims to optimize.

4 Problem Formulation

We focus on the problem of aligning a given prior policy with human behavior preferences by learning from *human intervention*. In this setting, a human expert observes a policy as it executes the task and intervenes whenever the policy behavior deviates from the expert's preference. The expert then continues executing the task until they are comfortable disengaging. Formally, we assume access to a prior policy π to execute, which is an optimal max-ent policy with respect to an unknown reward function r. We assume a human with an internal reward function r_{expert} that differs from r observes π's execution and provides interventions. The problem objective is to infer r_{expert} and use the inferred reward function to learn a policy $\hat{\pi}$ that matches the max-ent optimal policy with respect to r_{expert}. During learning, we can execute the updated policy under human supervision to collect new intervention samples. However, we want to minimize the number of samples collected, so as to limit the cognitive cost to the human. We assume access to a simulator.

If the ground truth r_{expert} were known, we could synthesize the max-ent optimal policy with respect to that reward using max-ent RL [46, 47]. We could then evaluate the success of a particular method by measuring how closely the learned policy $\hat{\pi}$ approximates this optimal policy. However, we cannot access the human's internal reward function in practice. Therefore, we assess the effectiveness of an approach by the human intervention rate during policy execution, measured as the fraction of time steps during which the human intervenes in a task episode. We aim to develop an algorithm to learn a policy that can minimize the number of intervention samples required to reach a target intervention rate threshold. Additionally, we design synthetic tests where we know the expert reward and train a max-ent policy under the ground-truth reward as a human proxy, so that we can directly measure the sub-optimality of the learned policy (see Sec. 6).

5 Max-Ent Residual-Q Inverse Reinforcement Learning (MEREQ)

In this section, we present MEREQ, a sample-efficient algorithm for alignment from human intervention. We first introduce a naive MaxEnt-IRL solution (Sec. 5.1), and analyze its drawbacks to motivate residual reward learning (Sec. 5.2). We then present the complete algorithm (Sec. 5.3).

5.1 A Naive Maximum-Entropy IRL Solution

A naive way to solve the target problem is to directly apply MaxEnt-IRL to infer the human reward function r_{expert} and find $\hat{\pi}$. We model the human expert with the widely recognized model of Boltzmann rationality [13, 14], which conceptualizes human intent through a reward function and portrays humans as choosing trajectories proportionally to their exponentiated rewards [49]. We model r_{expert} as a linear combination of features, as stated in Sec. 3.2. We initialize the learning policy $\hat{\pi}$ as the prior policy π. We then iteratively collect human intervention samples by executing $\hat{\pi}$, and then infer r_{expert} and update $\hat{\pi}$ based on the collected intervention samples. We refer to this solution as **MaxEnt-FT**, with FT denoting fine-tuning. In our experiments, we also study a variation with randomly initialized $\hat{\pi}$, denoted as **MaxEnt**.

In each sample collection iteration i, MaxEnt-FT executes the current policy $\hat{\pi}$ for T timesteps under human supervision. The single rollout of length T is split into two classes of segments depending on who takes control, which are policy segments ξ^{p}_1, ξ^{p}_2, …, ξ^{p}_m, and expert segments ξ^{e}_1, ξ^{e}_2, …, ξ^{e}_n, where a segment ξ is a sequence of state-action pairs $\xi = \{(\mathbf{s}_1, \mathbf{a}_1), \ldots, (\mathbf{s}_j, \mathbf{a}_j)\}$. We define the collected *policy trajectory* in this iteration as the union of all policy segments, $\Xi^{\text{p}} = \bigcup_{k=1}^{m} \xi^{\text{p}}_k$. Similarly, we define the *expert trajectory* as $\Xi^{\text{e}} = \bigcup_{k=1}^{n} \xi^{\text{e}}_k$. Note that $\sum_{k=1}^{m} |\xi^{\text{p}}_k| + \sum_{k=1}^{n} |\xi^{\text{e}}_k| = T$.

Under the Boltzmann rationality model, each expert segment follows the distribution in Eqn. (3). Assuming the expert segments are all independent from each other, the likelihood of the expert trajectory can be written as $p(\Xi^{\text{e}}|\theta) = \prod_{k=1}^{n} p(\xi^{\text{e}}_k|\theta)$. We can then infer the weights of the unknown human reward function by maximizing the likelihood of the observed expert trajectory, that is

$$\theta^\star = \arg\max_\theta \log p(\Xi^{\text{e}}|\theta) = \arg\max_\theta \sum_{k=1}^{n} \log p(\xi^{\text{e}}_k|\theta), \tag{5}$$

then update $\hat{\pi}$ to be the max-ent optimal policy with respect to the reward function $\theta^{\star\top}\mathbf{f}$. Note that directly optimizing these reward inference and policy update objectives completely disregards the prior policy. Thus, this naive solution is inefficient in the sense that it is expected to require many human interventions, as it overlooks the valuable information embedded in the prior policy.

5.2 Residual Reward Inference and Policy Update

In this work, we aim to develop an alternative algorithm that can utilize the prior policy to solve the target problem in a sample-efficient manner. We start with reframing the policy update step in the naive solution as a *policy customization* problem [15]. Specifically, we can rewrite the unknown human reward function as the sum of π's underlying reward function r and a *residual reward* function r_{R}. We expect some feature weights to be zero for r_{R}, specifically for the reward features for which the expert's preferences match those of the prior policy. Thus, we represent r_{R} as a linear combination of the non-zero weighted feature set $\mathbf{f}_{\mathrm{R}} : \mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}^{f_{\mathrm{R}}}$ with weights θ_{R}.

If θ_{R} is known, we can apply RQL to update the learning policy $\hat{\pi}$ without knowing r (see Sec. 3.1). Yet, θ_{R} is unknown, and MaxEnt can only infer the full reward weights θ (see Eqn. (5)). Instead, we introduce a novel method that enables us to *directly infer the residual weights θ_{R} from expert trajectories without knowing r*, and then apply RQL with π and r_R to update the policy $\hat{\pi}$, which will be more sample-efficient than the naive solution, MaxEnt. The residual reward inference method is derived as follows. By substituting the residual reward function into the maximum likelihood objective function, we obtain the following objective function:

$$\mathcal{L} = \sum_{k=1}^{n} \left[r(\xi_k^{\mathrm{e}}) + \theta_{\mathrm{R}}^{\top}\mathbf{f}_{\mathrm{R}}(\xi_k^{\mathrm{e}}) \right] - \log Z_k(\theta_{\mathrm{R}}), \tag{6}$$

where $\mathbf{f}_{\mathrm{R}}(\xi)$ is a shorthand for $\sum_{(\mathbf{s}_i,\mathbf{a}_i)\in\xi} \mathbf{f}_{\mathrm{R}}(\mathbf{s}_i,\mathbf{a}_i)$, and $r(\xi)$ is a shorthand for $\sum_{(\mathbf{s}_i,\mathbf{a}_i)\in\xi} r(\mathbf{s}_i,\mathbf{a}_i)$. The partition function Z_k is defined as $Z_k(\theta_{\mathrm{R}}) = \int p(\xi_k) \exp\left[r(\xi_k) + \theta_{\mathrm{R}}^{\top}\mathbf{f}_{\mathrm{R}}(\xi_k) \right] d\xi_k$, with $|\xi_k| = |\xi_k^{\mathrm{e}}|$ for each k. We can then derive the gradient of the objective function as:

$$\begin{aligned} \nabla_{\theta_{\mathrm{R}}}\mathcal{L} &= \sum_{k=1}^{n} \mathbf{f}_{\mathrm{R}}(\xi_k^{\mathrm{e}}) - \sum_{k=1}^{n} \frac{1}{Z_k(\theta_{\mathrm{R}})} \int p(\xi_k) \exp\left[r(\xi_k) + \theta_{\mathrm{R}}^{\top}\mathbf{f}_{\mathrm{R}}(\xi_k) \right] \mathbf{f}_{\mathrm{R}}(\xi_k) d\xi_k, \\ &= \sum_{k=1}^{n} \mathbf{f}_{\mathrm{R}}(\xi_k^{\mathrm{e}}) - \sum_{k=1}^{n} \mathbb{E}_{\xi_k \sim p(\xi_k|\theta_{\mathrm{R}})} \left[\mathbf{f}_{\mathrm{R}}(\xi_k) \right]. \end{aligned} \tag{7}$$

The second term is essentially the expectation of the feature counts of $\mathbf{f}_{\mathrm{R}}$ under the soft optimal policy under the current θ_{R}. Therefore, we approximate the second term with samples obtained by rolling out the current policy $\hat{\pi}$ in the simulation environment:

$$\sum_{k=1}^{n} \mathbb{E}_{\xi_k \sim p(\xi_k|\theta_{\mathrm{R}})} \left[\mathbf{f}_{\mathrm{R}}(\xi_k) \right] \approx \frac{1}{T} \sum_{k=1}^{n} |\xi_k^{\mathrm{e}}| \cdot \mathbb{E}_{\xi \sim \hat{\pi}(\xi)} \left[\mathbf{f}_{\mathrm{R}}(\xi) \right]. \tag{8}$$

The term $|\xi_k^{\mathrm{e}}|/T$ is introduced to match the rollout length with that of the expert intervention samples. We can then apply gradient descent to infer θ_{R} directly, without inferring the prior reward term r.

5.3 Algorithm

The complete MEREQ algorithm is shown in Algorithm 1. In summary, MEREQ consists of an outer loop for sample collection and an inner loop for policy updates. In each sample collection iteration i, MEREQ runs the current policy $\hat{\pi}$ under the supervision of a human expert, collecting policy trajectory Ξ_i^{p} and expert trajectory Ξ_i^{e} (Line 3). Afterward, MEREQ enters the inner policy update loop to update the policy using the collected samples, *i.e.*, Ξ_i^{p} and Ξ_i^{e}, during which the policy is rolled out in a simulation environment to collect samples for reward gradient estimation and policy training. Concretely, each policy update iteration j alternates between applying a gradient descent step with step-size η to update the residual reward weights θ_{R} (Line 10), where the gradient is estimated (Line 7) following Eqn. (7) and Eqn. (8), and applying RQL to update the policy using π and the updated θ_{R} (Line 11). The inner loop is terminated when the residual reward gradient is smaller than a certain threshold ϵ (Line 8-9). The outer loop is terminated when the expert intervention rate, denoted by λ, hits a certain threshold δ (Line 4-5).

Pseudo Expert Trajectories. Inspired by previous learning from intervention algorithms [31, 43], we further categorize the policy trajectory Ξ_i^{p} into *snippets* labeled as "good-enough" samples and "bad" samples. Let ξ represent a single continuous segment within Ξ_i^{p}, and let $[a, b) \circ \xi$ denote a *snippet* of the segment ξ, where $a, b \in [0, 1]$, $a \le b$, referring to the snippet starting from the $[a|\xi|]$ timestep to the $[b|\xi|]$ timestep of the segment. The absence of intervention in the initial portion of ξ implicitly indicates that the expert considers these actions satisfactory, leading us to classify the first $1 - \kappa$ fraction of ξ as "good-enough" samples. We aggregate all such "good-enough" samples to form what we term the *pseudo-expert* trajectory, defined as $\Xi_i^{+} := \{(\mathbf{s}, \mathbf{a}) | (\mathbf{s}, \mathbf{a}) \in [0, 1-\kappa) \circ \xi,\ \forall \xi \subset \Xi_i^{\mathrm{p}}\}$. Pseudo-expert samples offer insights into expert preferences without additional interventions. If MEREQ uses the pseudo-expert trajectory to learn the residual reward function, it is concatenated with the expert trajectory, resulting in an augmented expert trajectory set, $\Xi_i^{\mathrm{e}} = \Xi_i^{\mathrm{e}} \cup \Xi_i^{+}$, to replace the original expert trajectory. Adding these pseudo-expert samples only affects the gradient estimation step in Line 8 of Algorithm 1.

Algorithm 1 Learn Residual Reward Weights θ_{R} in MEReQ-IRL Framework

```
Require: $\pi$, $\delta$, $\epsilon$, $\mathbf{f}_{\mathrm{R}}$, and $\eta$
 1: $\theta_{\mathrm{R}} \leftarrow \mathbf{0}$, $\hat{\pi} \leftarrow \pi$
 2: for $i = 0, \ldots, N_{\text{data}}$ do
 3:     Execute current policy $\hat{\pi}$ under expert supervision to get $\Xi_i^{\mathrm{e}}$ and $\Xi_i^{\mathrm{p}}$
 4:     if $\lambda_i = \mathtt{len}(\Xi_i^{\mathrm{e}})/\mathtt{len}(\Xi_i^{\mathrm{p}} + \Xi_i^{\mathrm{e}}) < \delta$ then        ▷ Intervention rate lower than threshold
 5:         return
 6:     for $j = 0, \ldots, N_{\text{update}}$ do
 7:         Estimate the residual reward gradient $\nabla_{\theta_{\mathrm{R}}}\mathcal{L}$
 8:         if $\nabla_{\theta_{\mathrm{R}}}\mathcal{L} < \epsilon$ then        ▷ $\theta_{\mathrm{R}}$ converges
 9:             return
10:         $\theta_{\mathrm{R}} \leftarrow \theta_{\mathrm{R}} + \eta\nabla_{\theta_{\mathrm{R}}}\mathcal{L}$
11:         $\hat{\pi} \leftarrow \mathtt{Residual\_Q\_Learning}(\pi, \hat{\pi}, \mathbf{f}_{\mathrm{R}}, \theta_{\mathrm{R}})$
```

6 Experiments

Tasks. We design multiple simulated and real-world tasks, which are categorized into two settings depending on the expert type. First, we consider learning from a *synthesized* expert. We specify a residual reward function and train an expert policy. Then, we define a *heuristic-based* rule to decide when the expert should intervene or disengage. Since we know the expert policy, we can directly evaluate the sub-optimality of the learned policy. Under this setting, we consider four simulated tasks: 1) ***Highway-Sim:*** The task is to control a vehicle to navigate through highway traffic in the `highway-env` [50]. The prior policy can change lanes arbitrarily to maximize progress, while the expert policy encourages the vehicle to stay in the right-most lane; 2) ***Bottle-Pushing-Sim:*** The task is to control a robot arm to push a wine bottle to a goal position in `MuJoCo` [51]. The prior policy can push the bottle anywhere along the height of the bottle, while the expert policy encourages pushing near the bottom of the bottle; 3) ***Erasing-Sim:*** In this task, a robot arm erases a marker on a whiteboard in `MuJoCo` [51]. The prior policy applies insufficient force for effective erasing, whereas the expert encourages greater contact force to ensure the marker is fully erased; 4) ***Pillow-Grasping-Sim:*** The task is to control a robot arm to grasp a pillow in `MuJoCo` [51]. The prior policy does not have a grasping point preference, whereas the expert favors grasping from the center.

We then validate MEREQ with *human-in-the-loop* (HITL) experiments. The tasks are similar to the ones with synthesized experts, specifically: 1) ***Highway-Human:*** A human expert monitoring task execution through a GUI and intervening using a keyboard. The human is instructed to keep the vehicle in the rightmost lane if possible; 2) ***Bottle-Pushing-Human:*** This experiment is conducted on a Fanuc LR Mate 200*i*D/7L 6-DoF robot arm with a customized tooltip to push the bottle. The

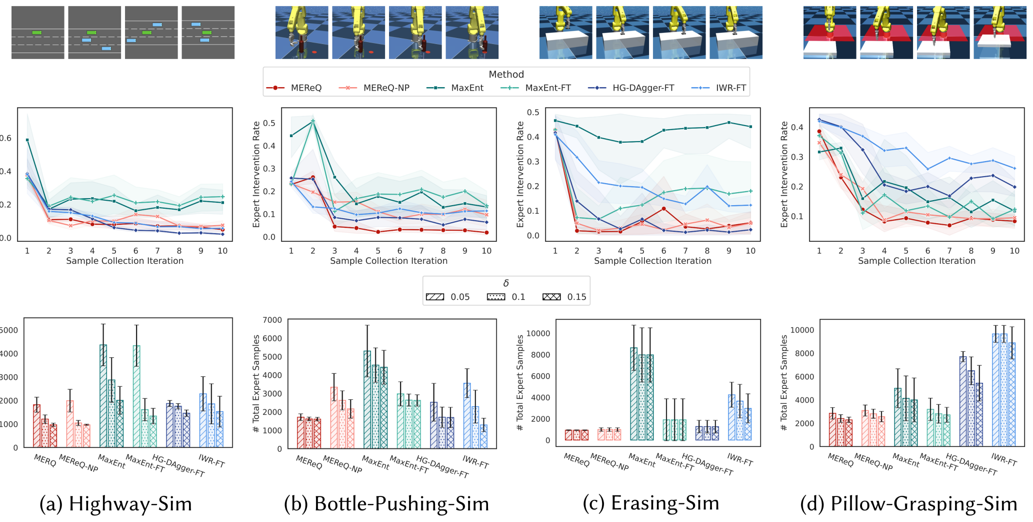


Figure 2: **Sample Efficiency. (Top) MEReQ** converges faster and maintains at low intervention rate throughout the sample collection iterations. The error bands indicate a 95% confidence interval across 8 trials. **(Bottom) MEReQ** requires fewer total expert samples to achieve comparable policy performance compared to baselines under varying intervention rate thresholds δ. The error bars indicate a 95% confidence interval. See Tab. 1 in Appendix B for detailed values.

human expert intervenes with a SpaceMouse when the robot does not aim for the bottom of the bottle; 3) ***Pillow-Grasping-Human:*** The experiment configuration is similar to bottle pushing, but the robot arm is equipped with a two-finger gripper. In these experiments, the specific algorithm variant was hidden from the expert during each trial. Please refer to Appendix A for more details.

Baselines and Evaluation Protocol. We compare **MEReQ** with the following baselines: **MEReQ-NP**, a MEReQ variation that does not use pseudo-expert trajectories (*i.e.*, **N**o **P**seudo); 2) **MaxEnt-FT**, the naive max-ent IRL solution (see Sec. 5.1); 3) **MaxEnt**, the naive solution but with random policy initialization; 4) **HG-DAgger-FT**, a variant of DAgger tailored for interactive imitation learning (IL) from human experts in real-world systems [4]; 5) **IWR-FT**, an intervention-based behavior cloning method with intervention weighted regression [31]. To ensure a fair comparison between **MEReQ** and the two interactive IL methods, we implemented the following adaptations: 1) We rolled out the prior policy to collect samples, which were then used to warm start **HG-DAgger-FT** and **IWR-FT** with behavior cloning. As shown in Fig. 2 (top), the initial intervention rates of the warm-started **HG-DAgger-FT** and **IWR-FT** are comparable to those of the prior policy of **MEReQ**; 2) Since both interactive IL methods maintain a dataset of all collected expert samples, we retained the full set of expert trajectories from each iteration, $\Xi^{\mathrm{e}} = \bigcup_i \Xi_i^{\mathrm{e}}$, where i denotes the iteration number, for the residual reward gradient calculation (Algorithm 1, line 7) of **MEReQ**. As discussed in Sec. 4, we use expert intervention rate as the main criterion to assess policy performance. We are primarily interested in the *sample efficiency* of the tested approaches. Specifically, we measure the number of expert samples required to have the intervention rate λ reach a certain threshold value δ.

6.1 Experimental Results

Experiments with Synthesized Experts. We evaluate each method using 8 random seeds and 10 data collection iterations per run. For each method, we compute the number of expert intervention samples needed to reach intervention rate thresholds $\delta = [0.05, 0.1, 0.15]$. As shown in Fig. 2 (top), **MEReQ** consistently achieves higher sample efficiency than baselines across all tasks and thresholds. Notably, it exhibits significantly lower variance across seeds compared to **HG-DAgger-FT** and **IWR-FT**, especially in more challenging environments like *Bottle-Pushing-Sim*, *Erasing-Sim*, and *Pillow-Grasping-Sim*. We further analyze behavior alignment in *Bottle-Pushing-Sim*, and

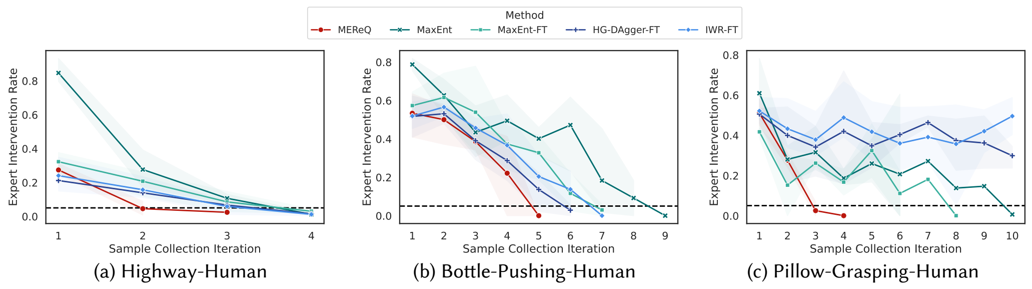


Figure 3: **Human Effort.** **MEReQ** can effectively reduce human efforts. The error bands indicate a 95% confidence interval across 3 trials. See Tab. 2 in Appendix B for detailed values.

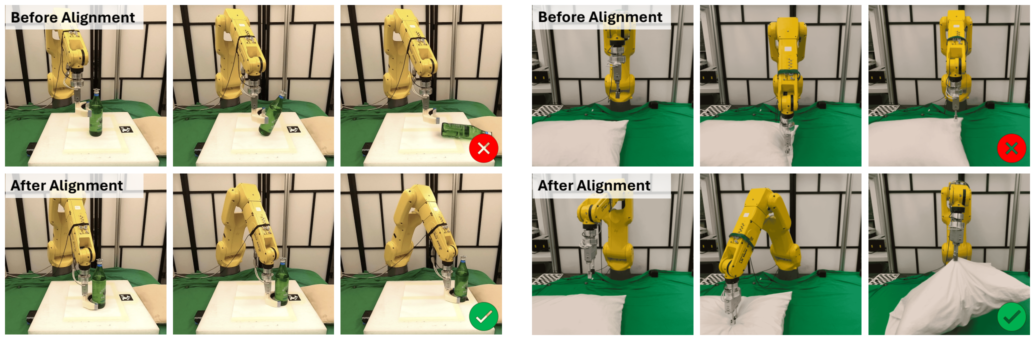


Figure 4: **Left:** ***Bottle-Pushing-Human*** **Rollout.** Before alignment, the robot knocks down the bottle with a high contact point. The robot pushes the bottle to the goal position with low contact point after alignment. **Right:** ***Pillow-Grasping-Human*** **Rollout.** Before alignment, the robot fails to grasp the pillow by the center. The robot grasps the pillow successfully after alignment.

find that the **MEReQ** policy matches the synthesized expert more closely in terms of feature and reward distributions than baselines (see Appendix B for more detailed results).

We would like to highlight two design choices that enable **MEReQ**'s sample efficiency and stability. First, the combination of residual reward learning and RQL allows MEReQ to effectively leverage the prior policy to facilitate sample efficiency, thus outperforming **MaxEnt-FT**. In particular, **MaxEnt-FT**'s expert intervention rate quickly rises to match that of **MaxEnt** after the first iteration in *Bottle-Pushing-Sim*, suggesting it only benefits from the prior policy in the early stage. Second, incorporating pseudo-expert samples further stabilizes training. **MEReQ** shows lower variance than **MEReQ-NP**. We hypothesize this because expert sample variance increases when intervention rates are low. Pseudo-expert samples help reduce the variance and stabilize training.

Human-in-the-loop Experiments. We investigate whether **MEReQ** effectively reduces the effort required from human experts. We set $\delta = 0.05$ and conducted three trials per method with a human expert. In *Highway-Human*, the human expert is tasked with supervising 50 rollouts, each consisting of 40 steps, during every outer loop. For both *Bottle-Pushing-Human* and *Pillow-Grasping-Human*, 10 rollouts are supervised by the expert in each outer loop. The training process concludes once the specified threshold is reached. As shown in Fig. 3, compared to the max-ent IRL baselines, **MEReQ** aligns the prior policy with human preferences in fewer sample collection iterations and with fewer human intervention samples (See Tab. 2 in Appendix B). These results demonstrate that **MEReQ** can be effectively adopted in real-world applications. As shown in Fig. 4, the prior policies fail to complete the *Bottle-Pushing* or *Pillow-Grasping* tasks due to inappropriate contact points, while the aligned policies successfully complete the tasks after adaptation from human interventions.

7 Limitations and Future Work

We introduce MEReQ, a novel algorithm for sample-efficient policy alignment from human intervention, which learns a residual reward function capturing the discrepancy between the human expert's and the prior policy's rewards. Across seven tasks in both simulation and real-world systems, MEReQ achieves alignment with significantly fewer human interventions than baseline approaches. While these results highlight the effectiveness of MEReQ, several limitations and promising future directions remain.

First, the current policy-updating process requires rollouts in a simulation environment, causing delays between sample-collection iterations. Parallel rollouts could speed up the training process. Adopting offline or model-based RL could also be a promising direction. Second, high variance in expert intervention samples could perturb the stability of MEREQ's training procedure. While the pseudo-expert approach can mitigate this issue, it is nevertheless a heuristic. More principled methods to reduce sample variance may be useful to further improve MEREQ. Additionally, noise and inconsistency in intervention actions may also perturb performance. We report preliminary studies on these effects across different algorithms in App. B, though how to fully address them is beyond the current scope and remains an important avenue for future research.

Third, MEREQ follows the linear reward model commonly used in inverse reinforcement learning (IRL). We are actively exploring IRL methods without this assumption [52, 53] and plan to extend MEReQ along this line in future work. Finally, in our human-in-the-loop experiments, each task was overseen by a single operator, which may introduce bias based on that person's skills, system familiarity, and tolerance level to undesirable behaviors. A broader study involving more participants would deepen our insight into how trust and subjectivity influence the timing, criteria, and frequency of interventions.

References

[1] J. Ji, T. Qiu, B. Chen, B. Zhang, H. Lou, K. Wang, Y. Duan, Z. He, J. Zhou, Z. Zhang, et al. Ai alignment: A comprehensive survey. *arXiv preprint arXiv:2310.19852*, 2023.

[2] C. Arzate Cruz and T. Igarashi. A survey on interactive reinforcement learning: Design principles and open challenges. In *Proceedings of the 2020 ACM designing interactive systems conference*, pages 1195–1209, 2020.

[3] Y. Cui, P. Koppol, H. Admoni, S. Niekum, R. Simmons, A. Steinfeld, and T. Fitzgerald. Understanding the relationship between interactions and outcomes in human-in-the-loop machine learning. In *International Joint Conference on Artificial Intelligence*, 2021.

[4] M. Kelly, C. Sidrane, K. Driggs-Campbell, and M. J. Kochenderfer. Hg-dagger: Interactive imitation learning with human experts. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 8077–8083. IEEE, 2019.

[5] H. Liu, S. Nasiriany, L. Zhang, Z. Bao, and Y. Zhu. Robot learning on the job: Human-in-the-loop autonomy and learning during deployment. *Robotics: Science and Systems (R:SS)*, 2023.

[6] J. Zhang and K. Cho. Query-efficient imitation learning for end-to-end autonomous driving. *arXiv e-prints*, pages arXiv–1605, 2016.

[7] S. Ross and D. Bagnell. Efficient reductions for imitation learning. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 661–668. JMLR Workshop and Conference Proceedings, 2010.

[8] D. Garg, S. Chakraborty, C. Cundy, J. Song, and S. Ermon. Iq-learn: Inverse soft-q learning for imitation. *Advances in Neural Information Processing Systems*, 34:4028–4039, 2021.

[9] Y. Jiang, C. Wang, R. Zhang, J. Wu, and L. Fei-Fei. Transic: Sim-to-real policy transfer by learning from online correction. *arXiv preprint arXiv:2405.10315*, 2024.

[10] A. NG. Algorithms for inverse reinforcement learning. In *Proc. of 17th International Conference on Machine Learning, 2000*, pages 663–670, 2000.

[11] B. D. Ziebart, A. Maas, J. A. Bagnell, and A. K. Dey. Maximum entropy inverse reinforcement learning. In *Proceedings of the 23rd national conference on Artificial intelligence-Volume 3*, pages 1433–1438, 2008.

[12] S. Arora and P. Doshi. A survey of inverse reinforcement learning: Challenges, methods and progress. *Artificial Intelligence*, 297:103500, 2021.

[13] J. Von Neumann and O. Morgenstern. *Theory of games and economic behavior, 2nd rev*. Princeton university press, 1947.

[14] C. L. Baker, J. B. Tenenbaum, and R. R. Saxe. Goal inference as inverse planning. In *Proceedings of the annual meeting of the cognitive science society*, volume 29, 2007.

[15] C. Li, C. Tang, H. Nishimura, J. Mercat, M. Tomizuka, and W. Zhan. Residual q-learning: Offline and online policy customization without value. *Advances in Neural Information Processing Systems*, 36, 2024.

[16] Y. Yue, J. Broder, R. Kleinberg, and T. Joachims. The k-armed dueling bandits problem. *Journal of Computer and System Sciences*, 78(5):1538–1556, 2012.

[17] A. Jain, B. Wojcik, T. Joachims, and A. Saxena. Learning trajectory preferences for manipulators via iterative improvement. *Advances in neural information processing systems*, 26, 2013.

[18] P. F. Christiano, J. Leike, T. Brown, M. Martic, S. Legg, and D. Amodei. Deep reinforcement learning from human preferences. *Advances in neural information processing systems*, 30, 2017.

[19] E. Bıyık, D. P. Losey, M. Palan, N. C. Landolfi, G. Shevchuk, and D. Sadigh. Learning reward functions from diverse sources of human feedback: Optimally integrating demonstrations and preferences. *The International Journal of Robotics Research*, 41(1):45–67, 2022.

[20] K. Lee, L. Smith, and P. Abbeel. Pebble: Feedback-efficient interactive reinforcement learning via relabeling experience and unsupervised pre-training. In *38th International Conference on Machine Learning, ICML 2021*. International Machine Learning Society (IMLS), 2021.

[21] X. Wang, K. Lee, K. Hakhamaneshi, P. Abbeel, and M. Laskin. Skill preferences: Learning to extract and execute robotic skills from human feedback. In *Conference on Robot Learning*, pages 1259–1268. PMLR, 2022.

[22] L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray, et al. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744, 2022.

[23] V. Myers, E. Bıyık, and D. Sadigh. Active reward learning from online preferences. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7511–7518. IEEE, 2023.

[24] R. Rafailov, A. Sharma, E. Mitchell, C. D. Manning, S. Ermon, and C. Finn. Direct preference optimization: Your language model is secretly a reward model. *Advances in Neural Information Processing Systems*, 36, 2024.

[25] J. Hejna, R. Rafailov, H. Sikchi, C. Finn, S. Niekum, W. B. Knox, and D. Sadigh. Contrastive preference learning: Learning from human feedback without reinforcement learning. In *The Twelfth International Conference on Learning Representations*, 2023.

[26] T. Tian, C. Xu, M. Tomizuka, J. Malik, and A. Bajcsy. What matters to you? towards visual representation alignment for robot learning. In *The Twelfth International Conference on Learning Representations*, 2023.

[27] W. Saunders, G. Sastry, A. Stuhlmüller, and O. Evans. Trial without error: Towards safe reinforcement learning via human intervention. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, pages 2067–2069, 2018.

[28] Z. Wang, X. Xiao, B. Liu, G. Warnell, and P. Stone. Appli: Adaptive planner parameter learning from interventions. In *2021 IEEE international conference on robotics and automation (ICRA)*, pages 6079–6085. IEEE, 2021.

[29] C. Celemin and J. Ruiz-del Solar. An interactive framework for learning continuous actions policies based on corrective feedback. *Journal of Intelligent & Robotic Systems*, 95:77–97, 2019.

[30] Z. M. Peng, W. Mo, C. Duan, Q. Li, and B. Zhou. Learning from active human involvement through proxy value propagation. *Advances in neural information processing systems*, 36, 2024.

[31] A. Mandlekar, D. Xu, R. Martín-Martín, Y. Zhu, L. Fei-Fei, and S. Savarese. Human-in-the-loop imitation learning using remote teleoperation. *arXiv preprint arXiv:2012.06733*, 2020.

[32] J. Spencer, S. Choudhury, M. Barnes, M. Schmittle, M. Chiang, P. Ramadge, and S. Srinivasa. Learning from interventions: Human-robot interaction as both explicit and implicit feedback. In *16th Robotics: Science and Systems, RSS 2020*. MIT Press Journals, 2020.

[33] W. B. Knox and P. Stone. Reinforcement learning from human reward: Discounting in episodic tasks. In *2012 IEEE RO-MAN: The 21st IEEE international symposium on robot and human interactive communication*, pages 878–885. IEEE, 2012.

[34] B. D. Argall, E. L. Sauser, and A. G. Billard. Tactile guidance for policy refinement and reuse. In *2010 IEEE 9th International Conference on Development and Learning*, pages 7–12. IEEE, 2010.

[35] T. Fitzgerald, E. Short, A. Goel, and A. Thomaz. Human-guided trajectory adaptation for tool transfer. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*, pages 1350–1358, 2019.

[36] A. Bajcsy, D. P. Losey, M. K. O'malley, and A. D. Dragan. Learning robot objectives from physical human interaction. In *Conference on robot learning*, pages 217–226. PMLR, 2017.

[37] A. Najar, O. Sigaud, and M. Chetouani. Interactively shaping robot behaviour with unlabeled human instructions. *Autonomous Agents and Multi-Agent Systems*, 34(2):35, 2020.

[38] N. Wilde, E. Biyik, D. Sadigh, and S. L. Smith. Learning reward functions from scale feedback. In *5th Annual Conference on Robot Learning*, 2021.

[39] G. Warnell, N. Waytowich, V. Lawhern, and P. Stone. Deep tamer: Interactive agent shaping in high-dimensional state spaces. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

[40] J. MacGlashan, M. K. Ho, R. Loftin, B. Peng, G. Wang, D. L. Roberts, M. E. Taylor, and M. L. Littman. Interactive learning from policy-dependent human feedback. In *International conference on machine learning*, pages 2285–2294. PMLR, 2017.

[41] D. Brown, W. Goo, P. Nagarajan, and S. Niekum. Extrapolating beyond suboptimal demonstrations via inverse reinforcement learning from observations. In *International conference on machine learning*, pages 783–792. PMLR, 2019.

[42] Y. Cui and S. Niekum. Active reward learning from critiques. In *2018 IEEE international conference on robotics and automation (ICRA)*, pages 6907–6914. IEEE, 2018.

[43] J. Spencer, S. Choudhury, M. Barnes, M. Schmittle, M. Chiang, P. Ramadge, and S. Srinivasa. Expert intervention learning: An online framework for robot learning from explicit and implicit human feedback. *Autonomous Robots*, pages 1–15, 2022.

[44] A. Bobu, A. Bajcsy, J. F. Fisac, and A. D. Dragan. Learning under misspecified objective spaces. In *Conference on Robot Learning*, pages 796–805. PMLR, 2018.

[45] A. Bobu, M. Wiggert, C. Tomlin, and A. D. Dragan. Feature expansive reward learning: Rethinking human input. In *Proceedings of the 2021 ACM/IEEE International Conference on Human-Robot Interaction*, pages 216–224, 2021.

[46] T. Haarnoja, H. Tang, P. Abbeel, and S. Levine. Reinforcement learning with deep energy-based policies. In *International conference on machine learning*, pages 1352–1361. PMLR, 2017.

[47] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pages 1861–1870. PMLR, 2018.

[48] E. T. Jaynes. Information theory and statistical mechanics. *Physical review*, 106(4):620, 1957.

[49] A. Bobu, D. R. Scobee, J. F. Fisac, S. S. Sastry, and A. D. Dragan. Less is more: Rethinking probabilistic models of human behavior. In *Proceedings of the 2020 acm/ieee international conference on human-robot interaction*, pages 429–437, 2020.

[50] E. Leurent. An environment for autonomous driving decision-making. `https://github.com/eleurent/highway-env`, 2018.

[51] E. Todorov, T. Erez, and Y. Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ international conference on intelligent robots and systems*, pages 5026–5033. IEEE, 2012.

[52] S. Levine. Nonlinear inverse reinforcement learning with gaussian processes. *NeurIPS*, 2011.

[53] A. Boularias. Relative entropy inverse reinforcement learning. In *PMLR*, 2011.

[54] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

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

[56] M. Menéndez, J. Pardo, L. Pardo, and M. Pardo. The jensen-shannon divergence. *Journal of the Franklin Institute*, 334(2):307–318, 1997.

A Detailed Environment Settings

Tasks. We design a series of both simulated and real-world tasks featuring discrete and continuous action spaces to evaluate the effectiveness of MEREQ. These tasks are categorized into two experiment settings: 1) Learning from synthesized expert with heuristic-based intervention rules, and 2) human-in-the-loop (HITL) experiments.

A.1 Learning from Synthesized Expert with Heuristic-based Intervention

In order to directly evaluate the sub-optimality of the learned policy through MEREQ, we specify a residual reward function and train an expert policy using this residual reward function and the prior reward function. We then define a heuristic-based intervention rule to decide when the expert should intervene or disengage. In this experiment setting, we consider two simulation environments for the highway driving task and the robot manipulation task.

A.1.1 Highway-Sim

Overview. We adopt the `highway-env` [50] environment for this task. The ego vehicle must navigate traffic using discrete actions to control speed and change lanes. The expert policy prefers the ego vehicle to stay in the right-most lane of a three-lane highway. Expert intervention is based on KL divergence between the expert and learned policies: the expert steps in if there is a significant mismatch for several consecutive steps and disengages once the distributions align for a sufficient number of steps. Each episode lasts for 40 steps. The sample roll-out is shown in Fig. 5.

Rewards Design. In *Highway-Sim* there are 5 available discrete actions for controlling the ego vehicle: $\mathcal{A} = \{\mathbf{a}_{\texttt{lane\_left}}, \mathbf{a}_{\texttt{idle}}, \mathbf{a}_{\texttt{lane\_right}}, \mathbf{a}_{\texttt{faster}}, \mathbf{a}_{\texttt{slower}}\}$. Rewards are based on 3 features: $\mathbf{f} = \{\mathbf{f}_{\texttt{collision}}, \mathbf{f}_{\texttt{high\_speed}}, \mathbf{f}_{\texttt{right\_lane}}\}$, defined as follows:

- $\mathbf{f}_{\texttt{collision}} \in \{0, 1\}$: 0 indicates no collision, 1 indicates a collision with a vehicle.
- $\mathbf{f}_{\texttt{high\_speed}} \in [0, 1]$: This feature is 1 when the ego vehicle's speed exceeds 30 m/s, and linearly decreases to 0 for speeds down to 20 m/s.
- $\mathbf{f}_{\texttt{right\_lane}} \in \{0, 0.5, 1\}$: This feature is 1 for the right-most lane, 0.5 for the middle lane, and 0 for the left-most lane.

The reward is defined as a linear combination of the feature set with the weights θ. For the prior policy, we define the basic reward as

$$r = -0.5 * \mathbf{f}_{\texttt{collision}} + 0.4 * \mathbf{f}_{\texttt{high\_speed}}. \tag{A.1}$$

For the expert policy, we define its reward as the basic reward with an additional term on $\mathbf{f}_{\texttt{right\_lane}}$

$$\begin{aligned} r_{\text{expert}} = &-0.5 * \mathbf{f}_{\texttt{collision}} + 0.4 * \mathbf{f}_{\texttt{high\_speed}} \\ &+ 0.5 * \mathbf{f}_{\texttt{right\_lane}}. \end{aligned} \tag{A.2}$$

Both prior and expert policy are trained using Deep Q-Network (DQN) [54] with the reward defined above in Gymnasium [55] environment. The hyperparameters are shown in Tab. 6.

Intervention Rule. The expert intervention is determined by the KL divergence between the expert policy π_{e} and the learner policy $\hat{\pi}$ given the same state observation $\mathbf{s}$, denoted as $D_{\text{KL}}(\hat{\pi}(\mathbf{a}|\mathbf{s}) \parallel$

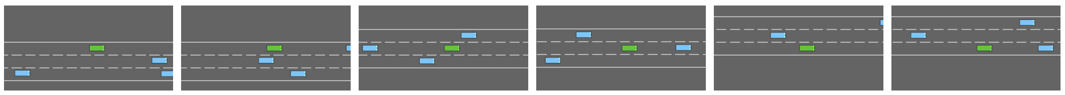


Figure 5: ***Highway-Sim* Sample Roll-out.** The green box is the ego vehicle, and the blue boxes are the surrounding vehicles. The bird-eye-view bounding box follows the ego vehicle.

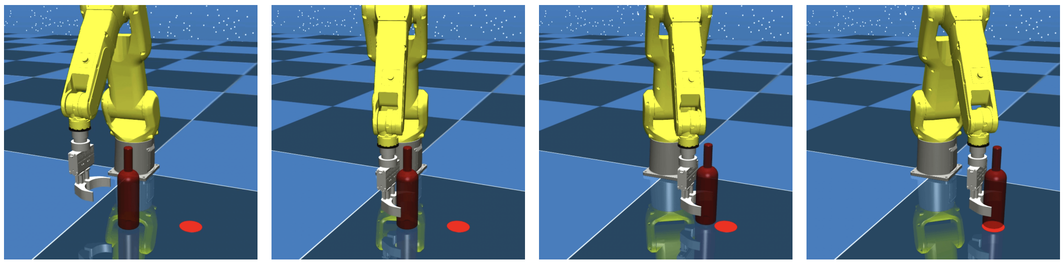


Figure 6: ***Bottle-Pushing-Sim* Sample Roll-out.** The location of the wine bottle and the goal are randomly initialized for each episode.

$\pi_{\mathrm{e}}(\mathbf{a}|\mathbf{s})$). At each time step, the state observation is fed into both policies to obtain the expert action $\mathbf{a}_{\mathrm{e}}$, the learner action $\hat{\mathbf{a}}$, and the expert action distribution $\pi_{\mathrm{e}}(\mathbf{a}|\mathbf{s})$, defined as

$$\pi_{\mathrm{e}}(\mathbf{a}|\mathbf{s}) = \frac{\exp(Q_{\mathrm{e}}^{\star}(\mathbf{s}, \mathbf{a}))}{\sum \exp(Q_{\mathrm{e}}^{\star}(\mathbf{s}, a_i))}, \tag{A.3}$$

where $Q_{\mathrm{e}}^{\star}$ is the soft Q-function. The learner's policy distribution $\hat{\pi}(\mathbf{a}|\mathbf{s})$ is treated as a *delta distribution* of the learner action $\delta[\mathbf{a}_{\mathrm{l}}]$.

We define heuristic thresholds $(D_{\mathrm{KL,upper}}, D_{\mathrm{KL,lower}}) = (1.62, 1.52)$. If the learner policy is in control and $D_{\mathrm{KL}} \geq D_{\mathrm{KL,upper}}$ for 2 consecutive steps, the expert policy takes over; During expert control, if $D_{\mathrm{KL}} \leq D_{\mathrm{KL,lower}}$ for 4 consecutive steps, the expert disengages. Each expert intervention must last at least 4 steps.

A.1.2 Bottle-Pushing-Sim

Overview. A 6-DoF robot arm is tasked with pushing a wine bottle to a random goal position. The expert policy prefers pushing from the bottom for safety. Expert intervention is based on state observation: the expert engages if the tooltip is too high, risking the bottle tilting for several consecutive steps, and disengages when the tooltip stays low enough for a sufficient number of steps. Each episode lasts for 100 steps. The sample roll-out is shown in Fig. 6.

Rewards Design. In *Bottle-Pushing-Sim*, the action space $\mathbf{a} \in \mathbb{R}^3$ is continuous, representing end-effector movements along the global x, y, and z axes. Each dimension ranges from -1 to 1, with positive values indicating movement in the positive direction and negative values indicating movement in the negative direction along the respective axes. All values are in centimeter. The rewards are based on 4 features: $\mathbf{f} = \{\mathbf{f}_{\texttt{tip2bottle}}, \mathbf{f}_{\texttt{bottle2goal}}, \mathbf{f}_{\texttt{control\_effort}}, \mathbf{f}_{\texttt{table\_distance}}\}$, with:

- $\mathbf{f}_{\texttt{tip2bottle}} \in [0, 1]$: This feature is 1 when the distance between the end-effector tool tip and the wine bottle's geometric center exceeds 30 cm, and decreases linearly to 0 as the distance approaches 0 cm.
- $\mathbf{f}_{\texttt{bottle2goal}} \in [0, 1]$: This feature is 1 when the distance between the wine bottle and the goal exceeds 30 cm, and decreases linearly to 0 as the distance approaches 0 cm.
- $\mathbf{f}_{\texttt{control\_effort}} \in [0, 1]$: This feature is 1 when the end-effector acceleration exceeds 5×10^{-3} m/s^2, and decreases linearly to 2 as the acceleration approaches 0.
- $\mathbf{f}_{\texttt{table\_distance}} \in [0, 1]$: This feature is 1 when the distance between the end-effector tool tip and the table exceeds 10 cm, and decreases linearly to 0 as the distance approaches 0 cm.

The reward is defined as a linear combination of the feature set with the weights θ. For the prior policy, we define the basic reward as

$$\begin{aligned} r = &-1.0 \, * \, \mathbf{f}_{\texttt{tip2bottle}} - 1.0 \, * \, \mathbf{f}_{\texttt{bottle2goal}} \\ &- 0.2 \, * \, \mathbf{f}_{\texttt{control\_effort}}. \end{aligned} \tag{A.4}$$

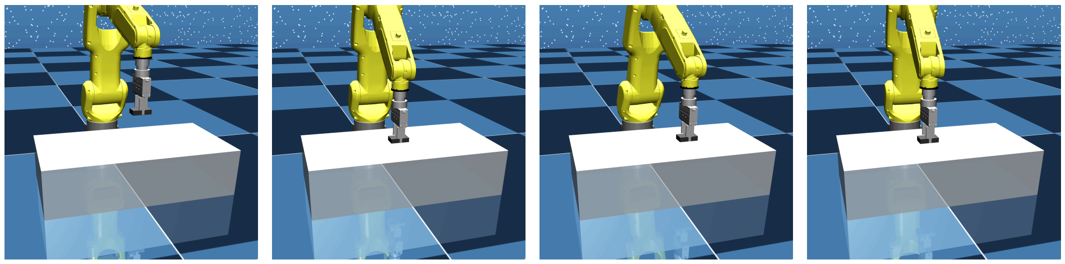


Figure 7: ***Erasing-Sim* Sample Roll-out.** The location of the whiteboard is fixed for each episode.

For the expert policy, we define the expert reward as the basic reward with an additional term on $\mathbf{f}_{\texttt{table\_distance}}$

$$r_{\text{expert}} = -1.0 * \mathbf{f}_{\texttt{tip2bottle}} - 1.0 * \mathbf{f}_{\texttt{bottle2goal}} - 0.2 * \mathbf{f}_{\texttt{control\_effort}} - 0.8 * \mathbf{f}_{\texttt{table\_distance}}. \tag{A.5}$$

Both prior and expert policy are trained using Soft Actor-Critic (SAC) [47] with the rewards defined above in MuJoCo [51] environment. The hyperparameters are shown in Tab. 8.

Intervention Rule. During learner policy execution, the expert policy takes over if either of the following conditions is met for 5 consecutive steps:

1. After 20 time steps, the bottle is not close to the goal ($\mathbf{f}_{\texttt{bottle2goal}} \geq 3$ cm) and the distance between the end-effector and the table exceeds 3 cm ($\mathbf{f}_{\texttt{table\_distance}} \geq 3$ cm).
2. After 40 time steps, the bottle is not close to the goal ($\mathbf{f}_{\texttt{bottle2goal}} \geq 3$ cm) and the bottle movement in the past time step is less than 0.1 cm.

During expert control, the expert disengages if either of the following conditions is met for 3 consecutive steps:

1. The distance between the end-effector and the table exceeds 3 cm ($\mathbf{f}_{\texttt{table\_distance}} \leq 3$ cm) and the bottle movement in the past time step is greater than 0.1 cm.
2. The bottle is close to the goal ($\mathbf{f}_{\texttt{bottle2goal}} \leq 3$ cm).

A.1.3 Erasing-Sim

Overview. A 6-DoF robot arm is tasked with erasing marker on a whiteboard on the table with an eraser. The expert policy prefers applying a larger normal force to ensure the erasing performance. Expert intervention is based on the contact force of the end-effector: the expert engages if the normal force applied by the end-effector is too small for several consecutive steps, and disengages after a fixed number of steps. Each episode lasts for 100 steps. The sample roll-out is shown in Fig. 7.

Rewards Design. In *Erasing-Sim*, the action space $\mathbf{a} \in \mathbb{R}^3$ is continuous, representing end-effector movements along the global x, y, and z axes. Each dimension ranges from -1 to 1, with positive values indicating movement in the positive direction and negative values indicating movement in the negative direction along the respective axes. All values are in centimeter. The rewards are based on 4 features: $\mathbf{f} = \{\mathbf{f}_{\texttt{tip\_hor\_move}}, \mathbf{f}_{\texttt{tip\_ver\_dist}}, \mathbf{f}_{\texttt{control\_effort}}, \mathbf{f}_{\texttt{tip\_force}}\}$, with:

- $\mathbf{f}_{\texttt{tip\_hor\_move}} \in [0, 1]$: This feature is 1 when the horizontal movement of the end-effector since last step exceeds 0.6 cm, and decreases linearly to 0 as the movement approaches 0.
- $\mathbf{f}_{\texttt{tip\_ver\_dist}} \in [0, 1]$: This feature is 1 when the distance between the eraser and the whiteboard exceeds 4 cm, and decreases linearly to 0 as the distance approaches 0 cm.
- $\mathbf{f}_{\texttt{control\_effort}} \in [0, 1]$: This feature is 1 when the end-effector acceleration exceeds 5×10^{-3} m/s^2, and decreases linearly to 2 as the acceleration approaches 0.

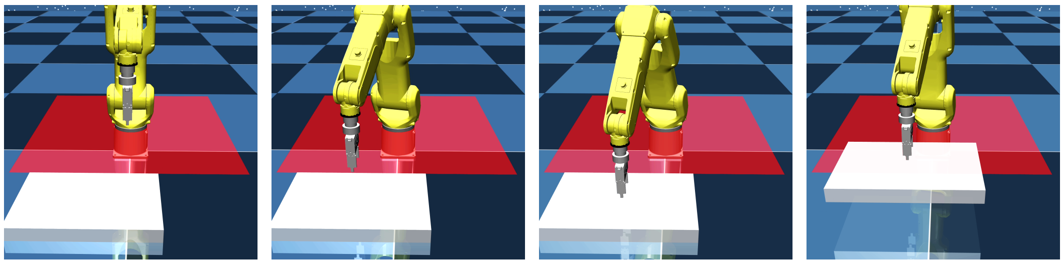


Figure 8: ***Pillow-Grasping-Sim* Sample Roll-out.** The expert prefers grasping from the center for improved success rate.

- $\mathbf{f}_{\texttt{tip\_force}} \in [0,1]$: This feature is 1 when the normal force applied by the eraser exceeds 4 N, and decreases linearly to 0 as the normal force approaches 0 N.

The reward is defined as a linear combination of the feature set with the weights θ. For the prior policy, we define the basic reward as

$$\begin{aligned} r = 1.0 * \mathbf{f}_{\texttt{tip\_hor\_move}} - 1.0 * \mathbf{f}_{\texttt{tip\_ver\_dist}} \\ - 0.2 * \mathbf{f}_{\texttt{control\_effort}}. \end{aligned} \tag{A.6}$$

For the expert policy, we define the expert reward as the basic reward with an additional term on $\mathbf{f}_{\texttt{table\_distance}}$

$$\begin{aligned} r_{\text{expert}} = 1.0 * \mathbf{f}_{\texttt{tip\_hor\_move}} - 1.0 * \mathbf{f}_{\texttt{tip\_ver\_dist}} \\ - 0.2 * \mathbf{f}_{\texttt{control\_effort}} + 2.0 * \mathbf{f}_{\texttt{tip\_force}}. \end{aligned} \tag{A.7}$$

Both prior and expert policy are trained using Soft Actor-Critic (SAC) [47] with the rewards defined above in MuJoCo [51] environment. The hyperparameters are shown in Tab. 8.

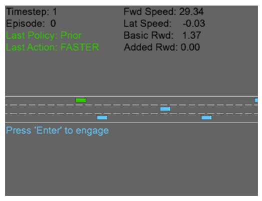
Intervention Rule. During learner policy execution, the expert policy takes over if the normal force applied by the end-effector is smaller than 2 N ($\mathbf{f}_{\texttt{tip\_force}} < 2$ N) for 5 consecutive steps. Expert control will last for 5 steps and automatically disengages.

A.1.4 Pillow-Grasping-Sim

Overview. A 6-DoF robot arm is tasked with grasping a pillow with a parallel two-finger gripper. The expert policy prefers grasping from the center. Expert intervention is based on the state observation: the expert engages if the gripper is not going lower and closer to the center, and disengages when the gripper is actively moving towards the center. Each episode lasts for 100 steps. The sample roll-out is shown in Fig. 7.

Rewards Design. In *Erasing-Sim*, the action space $\mathbf{a} \in \mathbb{R}^3$ is continuous, representing end-effector movements along the global x, y, and z axes. Each dimension ranges from -1 to 1, with positive values indicating movement in the positive direction and negative values indicating movement in the negative direction along the respective axes. All values are in centimeter. The gripper will automatically close once the end-effector reach the pillow. The rewards are based on 4 features: $\mathbf{f} = \{\mathbf{f}_{\texttt{tip2pillow}}, \mathbf{f}_{\texttt{pillow\_height}}, \mathbf{f}_{\texttt{control\_effort}}, \mathbf{f}_{\texttt{tip2center}}\}$, with:

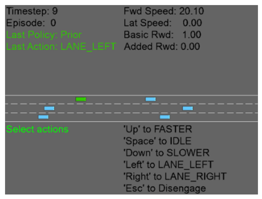
- $\mathbf{f}_{\texttt{tip2pillow}} \in [0,1]$: This feature is 1 when the vertical movement of the end-effector towards the surface of the pillow since last step exceeds 0.5 cm, and decreases linearly to 0 as the end-effector moving away from the surface of the pillow for more than 0.5 cm.
- $\mathbf{f}_{\texttt{pillow\_height}} \in [0,1]$: This feature is 1 when the distance between the pillow and the table surface 5 cm, and decreases linearly to 0 as the distance approaches 0 cm.
- $\mathbf{f}_{\texttt{control\_effort}} \in [0,1]$: This feature is 1 when the end-effector acceleration exceeds 5×10^{-3} m/s^2, and decreases linearly to 2 as the acceleration approaches 0.



```
Timestep: 1                Fwd Speed: 29.34
Episode:  0                Lat Speed:   -0.03
Last Policy: Prior         Basic Rwd:   1.37
Last Action: FASTER        Added Rwd: 0.00

Press 'Enter' to engage
```

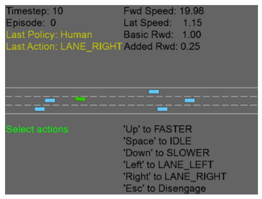
(a) Policy Control



```
Timestep: 9                Fwd Speed: 20.10
Episode:  0                Lat Speed:    0.00
Last Policy: Prior         Basic Rwd:   1.00
Last Action: LANE_LEFT     Added Rwd: 0.00

Select actions             'Up' to FASTER
                           'Space' to IDLE
                           'Down' to SLOWER
                           'Left' to LANE_LEFT
                           'Right' to LANE_RIGHT
                           'Esc' to Disengage
```

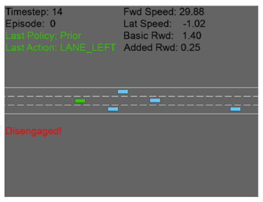
(b) Human Engage



```
Timestep: 10               Fwd Speed: 19.98
Episode:  0                Lat Speed:    1.15
Last Policy: Human         Basic Rwd:   1.00
Last Action: LANE_RIGHT    Added Rwd: 0.25

Select actions             'Up' to FASTER
                           'Space' to IDLE
                           'Down' to SLOWER
                           'Left' to LANE_LEFT
                           'Right' to LANE_RIGHT
                           'Esc' to Disengage
```

(c) Human Control



```
Timestep: 14               Fwd Speed: 29.88
Episode:  0                Lat Speed:   -1.02
Last Policy: Prior         Basic Rwd:   1.40
Last Action: LANE_LEFT     Added Rwd: 0.25

Disengaged!
```

(d) Human Disengage

Figure 9: ***Highway-Human* Graphic User Interface.** There are four different scenarios during the sample collection process. When the human expert engages and takes over the control, additional information would show up for available actions.

- $\mathbf{f}_{\texttt{tip2center}} \in [0, 1]$: This feature is 1 when the movement of the end-effector towards the center of the pillow since last step exceeds 0.5 cm, and decreases linearly to 0 as the end-effector moving away from the center of the pillow for more than 0.5 cm.

The reward is defined as a linear combination of the feature set with the weights θ. For the prior policy, we define the basic reward as

$$\begin{aligned} r = &-0.5 * \mathbf{f}_{\texttt{tip2pillow}} + 2.0 * \mathbf{f}_{\texttt{pillow\_height}} \\ &- 0.2 * \mathbf{f}_{\texttt{control\_effort}}. \end{aligned} \tag{A.8}$$

For the expert policy, we define the expert reward as the basic reward with an additional term on $\mathbf{f}_{\texttt{table\_distance}}$

$$\begin{aligned} r_{\text{expert}} = &-0.5 * \mathbf{f}_{\texttt{tip2pillow}} + 2.0 * \mathbf{f}_{\texttt{pillow\_height}} \\ &- 0.2 * \mathbf{f}_{\texttt{control\_effort}} - 0.8 * \mathbf{f}_{\texttt{tip2center}}. \end{aligned} \tag{A.9}$$

Both prior and expert policy are trained using Soft Actor-Critic (SAC) [47] with the rewards defined above in MuJoCo [51] environment. The hyperparameters are shown in Tab. 8.

Intervention Rule. During learner policy execution, the expert policy takes over if:

1. The horizontal movement of the end-effector towards the center of the pillow during last step is less than a pre-defined threshold for 5 consecutive steps. The threshold varies depending on the current vertical distance between the end-effector and the center. For vertical distance larger than 5 cm, the threshold is 0.4 cm; for vertical distance between 3 cm and 5 cm, the threshold is 0.2 cm; for vertical distance smaller than 3 cm, the threshold is -0.5 cm (moving away from the center for more than 0.5 cm in the last step).
2. The vertical movement of the end-effector towars the surface of the pillow during last step is less than 0.15 cm for 10 consecutive steps.

During expert control, the expert disengages if the horizontal end-effector towards the center of the pillow during last step is greater than the pre-defined threshold for 3 consecutive steps.

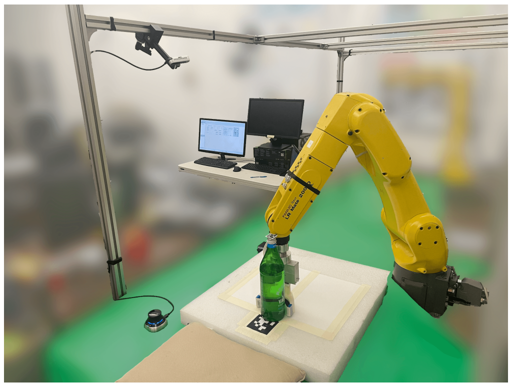
A.2 Human-in-the-loop Experiments

For the human-in-the-loop experiments, we substitute the synthesized experts in the corresponding experiments with human experts.

A.2.1 Highway-Human

Overview. We use the same `highway-env` environment with a customized Graphic User Interface (GUI) for human supervision. Human experts can intervene at will and control the ego vehicle using the keyboard. The sample GUI of 4 different scenarios are shown in Fig. 9.

Human Interface. We design a customized Graphic User Interface (GUI) for `highway-env` as shown in Fig. 9. The upper-left corner contains information about: 1) the step count in the current


(a) *Bottle-Pushing-Human* Hardware Setup

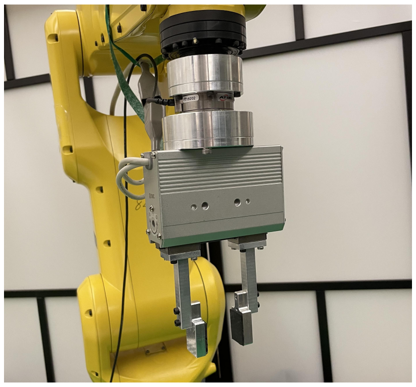

(b) *Pillow-Grasping-Human* Robot Gripper

Figure 10: **Hardware setups for robot experiments. (Left)** In the *Bottle-Pushing-Human* task, we use a Fanuc LR Mate 200iD/7L 6-DoF robot arm mounted on a tabletop, a fixed RealSense D435 depth camera for tracking AprilTags attached to the bottle and goal position, and a 3Dconnexion SpaceMouse for online human intervention. **(Right)** In the *Pillow-Grasping-Human* task, we use a two-finger parallel gripper mounted on the robot end-effector for grasping the pillow.

episode; 2) the total episode count; and 3) last executed action and last policy in control. The upper-right corner contains information about: 1) forward and lateral speed of the ego vehicle; and 2) basic and residual reward of the current state. The lower-left corner contains the user instruction on engaging and action selection. Whenever the human user is taking control, the lower-right corner shows the available actions and the corresponding keys.

A.2.2 Bottle-Pushing-Human

Overview. We use a Fanuc LR Mate 200iD/7L 6-DoF robot arm with a customized tooltip to push the bottle. Human experts can intervene at will and control the robot using a 3DConnexion SpaceMouse. Please refer to Fig. 4 (left) for a sample failure rollout where the robot knocks down the wine bottle before alignment, and a sample rollout where the robot successfully pushes the bottle to the goal position after alignment.

Human Interface. The hardware setup for the real-world experiment is shown in Fig. 10a. The robot arm is mounted on the tabletop. We use the RealSense d435 depth camera to track the April-Tags attached to the bottle and the goal position for the state feedback. The human expert uses the SpaceMouse to control the 3D position and orientation of the end-effector. The end-effector consists of a pair of tooltips specifically designed for the bottle-pushing task, which are 3D printed and attached to a parallel gripper with a fixed distance between the two fingers.

A.2.3 Pillow-Grasping-Human

We use the same robot arm with a standard two-finger parallel gripper (see Fig. 10b) to grasp the pillow. Human experts can intervene at will and control the robot using a 3DConnexion SpaceMouse. Please refer to Fig. 4 (right) for a sample failure roll-out where the robot fails to grasp the pillow by the center before alignment, and a sample roll-out where the robot successfully grasps the pillow by the center after alignment. The human interface is the same as *Bottle-Pushing-Human*.

B Additional Results

Sample Efficiency. Tab. 1 and Tab. 2 present the detailed numerical results corresponding to the plots shown in Fig. 2 and Fig. 3, respectively. Both tables report the mean values and 95% confidence intervals of the number of expert samples required by each algorithm. The results clearly demonstrate the advantage of MEREQ over the baseline methods with respect to sample efficiency.

Table 1: **MEReQ** and its variation **MEReQ-NP** require fewer total expert samples to achieve comparable policy performance compared to the max-ent IRL baselines **MaxEnt** and **MaxEnt-FT**, and interactive imitation learning baselines **HG-DAgger-FT** and **IWR-FT** under varying criteria strengths in different task and environment. Results are reported in `mean` (95%`ci`).

Environment	δ	**MEReQ**	**MEReQ-NP**	**MaxEnt**	**MaxEnt-FT**	**HG-DAgger-FT**	**IWR-FT**
Highway-Sim	0.05	**1819 (456)**	1990 (687)	4363 (1266)	4330 (1255)	1871 (183)	2284 (1039)
	0.1	**1208 (254)**	1043 (154)	2871 (1357)	1612 (673)	1754 (160)	1856 (1214)
	0.15	**965 (100)**	965 (37)	2005 (840)	1336 (468)	1458 (194)	1527 (930)
Bottle-Pushing-Sim	0.05	**1707 (261)**	3338 (1059)	5298 (2000)	2976 (933)	2519 (1459)	3554 (1118)
	0.1	**1613 (141)**	2621 (739)	4536 (1330)	2636 (468)	1706 (785)	2280 (1273)
	0.15	1604 (134)	2159 (717)	4419 (1306)	2618 (436)	1692 (787)	**1290 (516)**
Erasing-Sim	0.05	**925 (51)**	989 (228)	8627 (3019)	1899 (2796)	1268 (827)	4236 (1670)
	0.1	**923 (45)**	989 (228)	7965 (3610)	1899 (2796)	1258 (842)	3643 (2231)
	0.15	**923 (45)**	989 (228)	7965 (3610)	1899 (2796)	1258 (842)	2968 (1934)
Pillow-Grasping-Sim	0.05	**2848 (699)**	3086 (672)	4992 (2375)	3188 (1360)	7699 (624)	9645 (1034)
	0.1	**2398 (470)**	2807 (558)	4127 (2737)	2808 (1135)	6490 (1696)	9645 (1034)
	0.15	**2284 (332)**	2564 (633)	3993 (2681)	2715 (913)	5427 (2170)	8879 (1960)

Table 2: **MEReQ** require fewer total human samples to align the prior policy with human preference. Results are reported in `mean` (95%`ci`).

Environment	**MEReQ**	**MaxEnt**	**MaxEnt-FT**	**HG-DAgger-FT**	**IWR-FT**
Highway-Human	**654 (174)**	2482 (390)	1270 (440)	864 (194)	927 (237)
Bottle-Pushing-Human	**423 (107)**	879 (56)	564 (35)	450 (105)	524 (130)
Pillow-Grasping-Human	**149 (20)**	376 (123)	234 (141)	456 (126)	497 (301)

Behavior Alignment. As discussed in Sec.6.1, when using a synthesized expert, we can directly *measure the alignment between the behaviors of the learned and expert policies*, since both the expert policy distribution and the ground-truth expert reward are available. Specifically, for the *Bottle-Pushing-Sim* task, we collect sample rollouts from both policies, estimate their feature distributions, and compute the Jensen–Shannon divergence [56] between these distributions as a quantitative measure of behavior alignment. The feature distributions and their corresponding Jensen–Shannon divergences relative to the expert policy are shown in Fig. 11 and Tab. 3. We also visualize the reward distributions for all policies in Fig. 11 and report their means and standard deviations in Tab. 4. These results show that the MEReQ policy more closely matches the synthesized expert in terms of both feature and reward distributions compared to the baseline methods.

Performance under Noisy Intervention. Our work focuses on learning from interventions provided by a single, consistent human expert—*i.e.*, assuming a single trainer whose behavior preference does not shift. There could still be some noise, and indeed that was not controlled for in the experiments. However, handling noisy or inconsistent interventions is beyond our scope and remains a valuable direction for future work. As a preliminary exploration, we introduced Gaussian noise (mean 0, standard deviation 0.1) to the normalized actions $[-1, 1]$ of synthesized expert interventions (see Tab. 5, results are reported in `mean(95%ci)` with $\delta = 0.1$) in the *Bottle-Pushing-Sim* environment. While MEReQ's performance degrades under injected noise, it still outperforms the baselines.

C Implementation Details

In this section, we provide the hyperparameters for the prior policy training (see Tab. 6 and Tab. 8) and the Residual Q-Learning training (see Tab. 7 and Tab. 9).

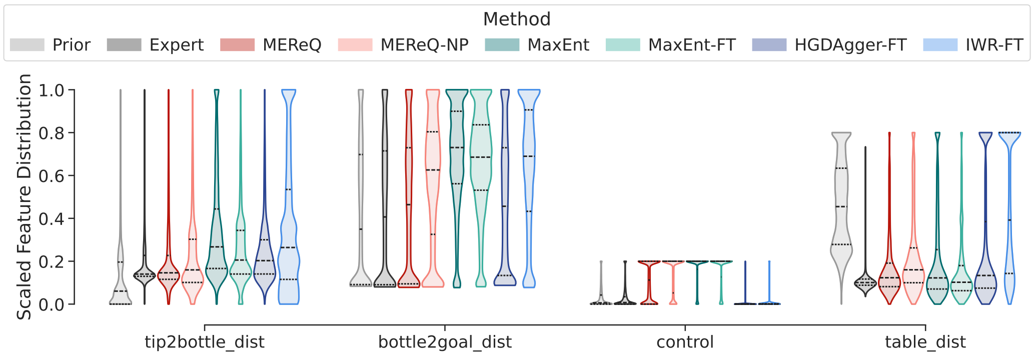


Figure 11: **Behavior Alignment.** We evaluate the policy distribution of all methods with a convergence threshold of 0.1 for each feature in the *Bottle-Pushing-Sim* environment. All methods align well with the **Expert** in the feature `table_dist` except for **IWR-FT**. Additionally, **MEReQ** aligns better with the **Expert** across the other three features compared to other baselines.

Table 3: The Jensen-Shannon Divergence of the feature distribution between each method and the synthesized expert in the *Bottle-Pushing-Sim* environment. Results are reported in `mean` (`95%ci`). The intervention rate threshold is set to 0.1.

Features	**MEReQ**	**MEReQ-NP**	**MaxEnt**	**MaxEnt-FT**	**HG-DAgger-FT**	**IWR-FT**
`scaled_tip2wine`	**0.237 (0.032)**	0.265 (0.023)	0.245 (0.022)	0.250 (0.038)	0.240 (0.017)	0.302 (0.058)
`scaled_wine2goal`	**0.139 (0.005)**	0.194 (0.044)	0.247 (0.046)	0.238 (0.039)	0.167 (0.033)	0.236 (0.040)
`scaled_eef_acc_sqrsum`	**0.460 (0.018)**	0.479 (0.022)	0.500 (0.026)	0.505 (0.016)	0.707 (0.006)	0.654 (0.022)
`scaled_table_dist`	**0.177 (0.021)**	0.219 (0.025)	0.236 (0.029)	0.210 (0.049)	0.284 (0.080)	0.308 (0.051)

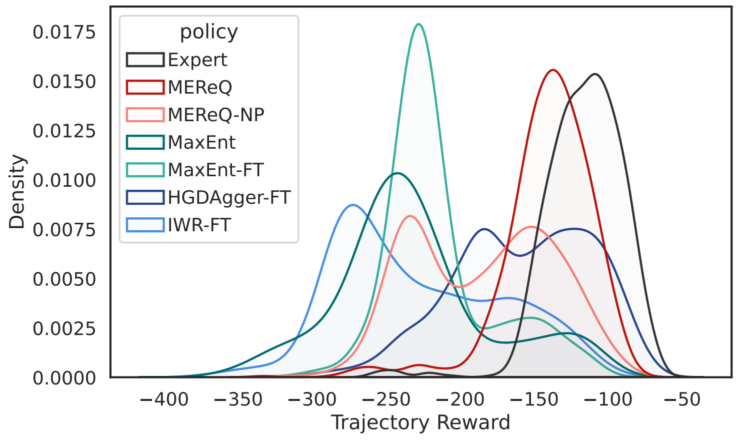


Figure 12: **Reward Alignment.** We visualize the reward distributions of all methods with a convergence threshold of 0.1 for each feature in the *Bottle-Pushing-Sim* environment. **MEReQ** aligns best with the **Expert** compared to other baselines.

Table 4: The mean and standard deviation of the reward distribution of each method.

Expert	**MEReQ**	**MEReQ-NP**	**MaxEnt**	**MaxEnt-FT**	**HG-DAgger-FT**	**IWR-FT**
-115.9 (25.9)	**-140.5 (30.8)**	-184.7 (46.9)	-231.1 (52.9)	-214.1 (36.7)	-157.5 (46.1)	-228.1 (56.1)

Table 5: Number of total expert samples with noisy intervention.

	MEReQ	MaxEnt-FT	HG-DAgger-FT	IWR-FT
No Noise	**1613 (141)**	2636 (468)	1706 (785)	2280 (1273)
10% Noise	**1043 (420)**	2228 (182)	3987 (1831)	11921 (1749)
50% Noise	**1011 (315)**	2612 (252)	3612 (1529)	11487 (3966)

Table 6: Hyperparameters of DQN Policies.

Hyperparameter	***Highway-Sim***	***Highway-Human***
`n_timesteps`	5×10^5	5×10^5
`learning_rate`	10^{-4}	10^{-4}
`batch_size`	32	32
`buffer_size`	1.5×10^4	1.5×10^4
`learning_starts`	200	200
`gamma`	0.8	0.8
`target_update_interval`	50	50
`train_freq`	1	1
`gradient_steps`	1	1
`exploration_fraction`	0.7	0.7
`net_arch`	$[256, 256]$	$[256, 256]$

Table 7: Hyperparameters of Residual DQN Policies.

Hyperparameter	***Highway-Sim***	***Highway-Human***
`n_timesteps`	4×10^4	4×10^4
`batch_size`	32	32
`buffer_size`	2000	2000
`learning_starts`	2000	2000
`learning_rate`	10^{-4}	10^{-4}
`gamma`	0.8	0.8
`target_update_interval`	50	50
`train_freq`	1	1
`gradient_steps`	1	1
`exploration_fraction`	0.7	0.7
`net_arch`	$[256, 256]$	$[256, 256]$
`env_update_freq`	1000	1000
`sample_length`	1000	1000
`epsilon`	0.03	0.03
`eta`	0.2	0.2

Table 8: Hyperparameters of SAC Policies.

Hyperparameter	***Bottle-Pushing-Sim***	***Bottle-Pushing-Human***	***Erasing-Sim***	***Pillow-Grasping-Sim***	***Pillow-Grasping-Human***
`n_timesteps`	5×10^4	5×10^4	5×10^4	5×10^4	5×10^4
`learning_rate`	5×10^{-3}	5×10^{-3}	5×10^{-3}	5×10^{-3}	5×10^{-3}
`batch_size`	512	512	512	512	512
`buffer_size`	10^6	10^6	10^6	10^6	10^6
`learning_starts`	5000	5000	5000	5000	5000
`ent_coef`	`auto`	`auto`	`auto`	`auto`	`auto`
`gamma`	0.9	0.9	0.9	0.9	0.9
`tau`	0.01	0.01	0.01	0.01	0.01
`train_freq`	1	1	1	1	1
`gradient_steps`	1	1	1	1	1
`net_arch`	$[400, 300]$	$[400, 300]$	$[400, 300]$	$[400, 300]$	$[400, 300]$

Table 9: Hyperparameters of Residual SAC Policies.

Hyperparameter	***Bottle-Pushing-Sim***	***Bottle-Pushing-Human***	***Erasing-Sim***	***Pillow-Grasping-Sim***	***Pillow-Grasping-Human***
`n_timesteps`	2×10^4	2×10^4	2×10^4	2×10^4	2×10^4
`batch_size`	512	512	512	512	512
`buffer_size`	10^6	10^6	10^6	10^6	10^6
`learning_starts`	5000	5000	5000	5000	5000
`learning_rate`	5×10^{-3}	5×10^{-3}	5×10^{-3}	5×10^{-3}	5×10^{-3}
`ent_coef`	`auto`	`auto`	`auto`	`auto`	`auto`
`ent_coef_prior`	0.035	0.035	0.035	0.035	0.035
`gamma`	0.9	0.9	0.9	0.9	0.9
`tau`	0.01	0.01	0.01	0.01	0.01
`train_freq`	1	1	1	1	1
`gradient_steps`	1	1	1	1	1
`net_arch`	$[400, 300]$	$[400, 300]$	$[400, 300]$	$[400, 300]$	$[400, 300]$
`env_update_freq`	1000	1000	1000	1000	1000
`sample_length`	1000	1000	2000	2000	1000
`epsilon`	0.2	0.2	0.1	0.1	0.4
`eta`	0.2	0.2	0.2	0.2	0.2