2026 IEEE International Conference on Robotics and Automation (ICRA 2026)
June 1-5, 2026. Vienna, Austria

CRED: Counterfactual Reasoning and Environment Design for Active Preference Learning

Yi-Shiuan Tung, Gyanig Kumar, Wei Jiang, Bradley Hayes, Alessandro Roncone
Department of Computer Science, University of Colorado Boulder
{yi-shiuan.tung, gyanig.kumar, wei.jiang, bradley.hayes, alessandro.roncone}@colorado.edu

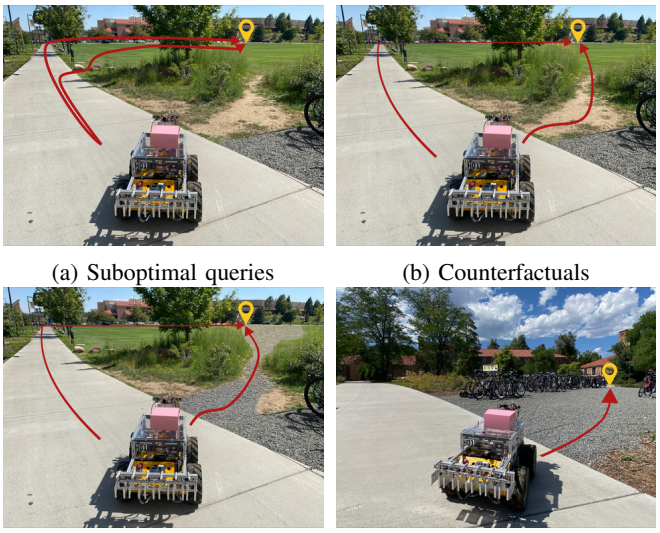
***Abstract*— As a robot's operational environment and tasks to perform within it grow in complexity, the explicit specification and balancing of optimization objectives to achieve a preferred behavior profile moves increasingly farther out of reach. These systems benefit strongly by being able to align their behavior to reflect human preferences and respond to corrections, but manually encoding this feedback is infeasible. Active preference learning (APL) learns human reward functions by presenting trajectories for ranking. However, existing methods sample from fixed trajectory sets or replay buffers that limit query diversity and often fail to identify informative comparisons. We propose CRED, a novel trajectory generation method for APL that improves reward inference by jointly optimizing environment design and trajectory selection to efficiently query and extract preferences from users. CRED "imagines" new scenarios through environment design and leverages counterfactual reasoning–by sampling possible rewards from its current belief and asking "What if this were the true preference?"–to generate trajectory pairs that expose differences between competing reward functions. Comprehensive experiments and a user study show that CRED significantly outperforms state-of-the-art methods in reward accuracy and sample efficiency and receives higher user ratings.**

I. INTRODUCTION

Robots deployed in the real world must operate in diverse, unpredictable environments and interact with users whose preferences and expectations vary widely. Predefining appropriate behaviors for all possible scenarios is infeasible–robot behavior must instead be fine-tuned through human feedback to ensure alignment with user expectations. For example, in autonomous delivery, different robot embodiments (e.g., wheeled vs. legged robots) may need to learn which terrains are acceptable to traverse based on both physical capabilities and user-specified trade-offs between efficiency and risk [1]. In domestic settings, users may prefer that household robots complete tasks more quickly, even if it means compromising slightly on task quality, such as folding laundry faster at the expense of producing perfectly neat folds. These types of trade-offs are often subtle, context dependent, and difficult to encode manually, underscoring the need for systems that can efficiently learn from user feedback during deployment.

Preference learning (PL) infers a reward function based on human-provided rankings of trajectories, eliminating the need to manually define rewards and allowing for personalization. Unlike inverse reinforcement learning, which depends on demonstration quality, PL allows non-expert users to provide input in complex domains such as Atari games and robot navigation [2], [3]. A key challenge, however, is that accurate reward learning often requires numerous preferences, limiting scalability to high-dimensional problems. To address this, active preference learning (APL) finds preference queries–sets of robot trajectories presented to a human–that maximize information gain [4], [5]. Prior work makes this optimization tractable by restricting the query set to pre-generated trajectories [6] or replay buffers [7], but this confines search to previously observed behaviors and may miss rare but informative scenarios needed to identify the true reward function.

We introduce CRED, a novel and efficient query generation method for APL that learns reward functions which generalize to different scenarios by using 1) **C**ounterfactual **R**easoning and 2) **E**nvironment **D**esign. CRED's counter-



(a) Suboptimal queries (b) Counterfactuals

(c) Environment design (d) User-aligned policy

Fig. 1: The delivery robot above (goal: yellow pin) optimizes its path by considering factors like travel time and terrain types. Through active preference learning, it infers human rewards from trajectory rankings. (a) However, current state-of-the-art methods often struggle to efficiently generate informative trajectory pairs for these queries, leading to suboptimal results. To overcome this, CRED incorporates two key contributions: (b) counterfactual reasoning, which explores varied hypothetical preferences to produce more diverse trajectories, and (c) environment design, which "imagines" different scenarios–e.g. altering terrain from grass to gravel–to enhance the system's generalization capabilities. (d) As a result, the robot aligns to human preferences when deployed.

factual reasoning generates queries that reflect different hypothesized human preferences instead of generating queries through random rollouts [6]. Assuming a linear human reward model $R(\xi) = w^T \Phi(\xi)$ (the dot product of reward weights w and trajectory features $\Phi(\xi)$), learning the reward function simplifies to learning w. We use Bayesian inference [8] to maintain a belief over w, updated with each human query. By sampling diverse w values (based on cosine similarity) from the current belief, CRED directly evaluates different human preferences and performs counterfactual reasoning, effectively asking "what if w_i or w_j were the true reward weight?" Our second key insight is that the environment itself strongly influences the generated trajectories and thus can unnecessarily bound the information value of queries. CRED employs Bayesian Optimization [9] to efficiently find *environment parameters* that yield the most informative preference queries.

The main contributions of this paper include a novel approach to preference learning, CRED, that leverages environment design and counterfactual reasoning. Empirical evaluations across close proximity (tabletop handover) and large length-scale (delivery navigation) tasks demonstrate that CRED's queries achieve higher reward accuracy and help users converge faster to the true reward function than prior methods. Moreover, user studies indicate that participants report lower mental workload and express stronger preference when interacting with CRED.

II. Related Works

Learning from Preferences. Preference learning infers a human's reward function by iteratively asking them to choose between robot trajectories [6]. In active preference learning (APL), the objective is to identify the most informative query by selecting the one that maximizes the expected difference between the prior and the posterior belief distributions over reward functions [4]. Subsequent work explores different query selection objectives: maximize mutual information of the query and the estimated weights [6], maximize regret [10], and maximize uncertainty in reward predictions [7]. For example, Lee et al. [7] proposes ensemble-based sampling, which selects trajectory pairs with high disagreement across an ensemble of learned reward models, and entropy-based sampling, which maximizes the entropy of the preference distribution. In our paper, we focus on maximizing mutual information, as it directly quantifies the expected reduction in uncertainty over the reward weights and aligns well with Bayesian preference learning. However, our framework is general and can be extended to support other objectives.

A major challenge of APL is generating trajectory pairs that effectively optimize the chosen query selection objective. Prior methods such as [6] and [7] address this by selecting queries from a fixed dataset or replay buffer, but this restricts the diversity of queries and does not scale well to long-horizon tasks. Moreover, existing approaches typically generate queries within a single, fixed environment, limiting their ability to explore informative regions of the feature space. In contrast, our method leverages counterfactual reasoning to synthesize trajectories from a belief distribution over reward weights and jointly optimizes environment parameters. This combination enables the generation of more informative and diverse preference queries.

Terrain Preferences for Navigation. Prior work has incorporated human terrain preferences into robot navigation tasks by learning which surfaces to traverse or avoid. Karnan et al. [11] learn a visual representation space to extrapolate preferences to out-of-distribution terrains, Mao et al. [1] train a bird's-eye view costmap from human feedback, and Zhang et al. [3] extend this by using demonstrations and counterfactuals to solicit additional human annotations. We adopt this domain for evaluation, but unlike these approaches which emphasize representation and costmap learning, our method targets the active selection of informative preference queries. As such, our framework can serve as a complementary front-end to generate the initial preference data that these methods require.

Environment Design in RL and Robotics. Environment design treats environment parameters as optimizable variables. In RL, it has been leveraged for curriculum learning to improve generalization and convergence, for instance, through co-evolution of agents and environment difficulty [12] or by modifying parameters to maximize an agent's learning potential [13]. In human-robot interaction, environment modification has been used to generate interpretable robot behaviors [14], legible human motion [15], [16], and to support collaborative teaming in settings like warehouse design [17] or tabletop reorganization [18]. Most relevant to our setting is the work of [19], which formulates environment design for inverse reinforcement learning (IRL) as a bilevel optimization problem over transition dynamics to elicit more informative demonstrations from experts. Our work adapts this idea to preference learning, where feedback comes from comparisons rather than demonstrations. We apply environment design within APL to identify informative environments that improve query informativeness and reduce posterior uncertainty over the reward function.

III. Preliminaries

Model. We consider a fully observable environment modeled as a Markov decision process (MDP) consisting of $\{S, A, T, R, \gamma, S_0\}$, where S is the set of states, A is the set of actions, $T : S \times A \times S \to [0,1]$ is the transition function, $R : S \times A \to \mathbb{R}$ is the reward function, $\gamma \in [0,1)$ is the discount factor, and S_0 is the initial state distribution. Let $s_t \in S$ and $a_t \in A$ denote the state and action at time step t. A trajectory $\xi \in \Xi$ is a finite sequence of state-action pairs: $\xi = ((s_0, a_0), (s_1, a_1), \ldots, (s_H, a_H))$ where H is the planning horizon. We overload the notation R and define the total reward of a trajectory as the sum of discounted per-step rewards: $R(\xi) = \sum_{t=0}^{H} \gamma^t R(s_t, a_t)$.

We assume a linear reward model over trajectory features: $R(\xi) = w^T \Phi(\xi)$ where $w \in \mathbb{R}^d$ are the unknown reward weights, and $\Phi(\xi) \in \mathbb{R}^d$ is the feature vector for trajectory ξ. To construct $\Phi(\xi)$, we define features over individual state-action pairs: let $\phi(s_t, a_t) \in \mathbb{R}^d$ be the feature vector

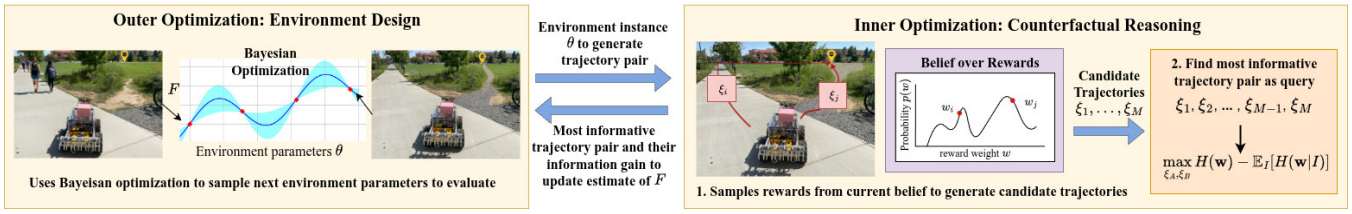


Fig. 2: System overview of CRED as a bilevel optimization problem. **Outer optimization (Environment Design):** Bayesian optimization selects environment parameters θ to evaluate, seeking those that maximize estimates of query informativeness F. **Inner optimization (Counterfactual Reasoning):** Given θ, the system samples reward weights from the current belief to generate candidate trajectories $\{\xi_1, \ldots, \xi_M\}$. The most informative trajectory pair is returned to the outer optimization.

at timestep t. The trajectory-level feature vector is then the discounted sum of per-step features: $\Phi(\xi) = \sum_{t=0}^{H} \gamma^t \phi(s_t, a_t)$. Once the reward weights w are learned from human feedback, we can optimize trajectories with respect to the learned reward function using reinforcement learning [20] or trajectory optimization methods such as CHOMP [21].

Preference learning. The objective of preference learning is to learn w by querying a human for their preferences between pairs of trajectories. A preference query typically asks "Do you prefer trajectory ξ_A or ξ_B?" [5]. If a human prefers ξ_A over ξ_B, it implies $R(\xi_A) > R(\xi_B)$, or equivalently $w^T\Phi(\xi_A) > w^T\Phi(\xi_B)$. From this strict inequality, we can derive that $w^T(\Phi(\xi_A) - \Phi(\xi_B)) > 0$. The human's preference I can be encoded by $I = sign(w^T(\Phi(\xi_A) - \Phi(\xi_B)))$.

Human input can exhibit variability due to uncertainty in expressed preferences, which can be modeled using Boltzmann rationality. Under this model, the likelihood of a preference (Eqn. 1) is given by a softmax function over the reward values of the queried trajectories.

$$P(I \mid \mathbf{w}) = \begin{cases} \frac{\exp(R(\xi_A))}{\exp(R(\xi_A))+\exp(R(\xi_B))} & \text{if } I = +1 \\ \frac{\exp(R(\xi_B))}{\exp(R(\xi_A))+\exp(R(\xi_B))} & \text{if } I = -1 \end{cases} \tag{1}$$

Let $p(w)$ be our current belief distribution of the reward weights. We can perform a Bayesian update to compute the posterior given human input I, $p(w|I) \propto p(I|w)p(w)$. For uniqueness, we constrain the norm of the reward weights such that $\|w\|_2 \leq 1$. Since $p(w)$ can have arbitrary shapes, we use an adaptive Metropolis algorithm [22] to learn the posterior distribution. While we can use domain knowledge to initialize a non-uniform prior over reward weights, we adopt a uniform prior in all of our experiments for generality. Based on [6], the algorithm presents the human with a preference query and updates the belief distribution of w until a fixed number of iterations is reached.

Active Synthesis of Preference Queries. To learn w efficiently using minimal queries, active learning methods select preference queries (ξ_A, ξ_B) that maximize information gain. This is equivalent to maximizing the mutual information between the query and the estimated weights w [6]. Our objective function f is

$$\max_{\xi_A, \xi_B} f(\xi_A, \xi_B) = \max_{\xi_A, \xi_B} H(\mathbf{w}) - \mathbb{E}_I[H(\mathbf{w}|I)] \tag{2}$$

where $H(w) = -\mathbb{E}_w[log(p(w))]$ is the information entropy of the belief $p(w)$. This objective finds preference queries such that the difference between the entropy of the prior and the posterior is maximized. However, directly optimizing this objective is challenging in practice: the search space is non-convex, and optimization often converges to local maxima, resulting in poor sample efficiency. Section IV discusses our approach of using counterfactual reasoning and environment design to more effectively generate trajectories that optimize this information gain objective.

Environment Design. In traditional preference learning, the environment is fixed. However, the transition dynamics determine which trajectories are feasible and which features are observable. Consequently, the informativeness of a preference query depends not just on the trajectories, but also on the environment in which they are embedded.

We formalize the problem of environment design for preference learning as the selection of a sequence of environment parameters $\theta \in \Theta$, where each θ defines a specific transition function $T_\theta \in \mathbb{T}$. These parameters capture aspects of the environment such as terrain types, obstacle placements, or object configurations, and they determine which trajectories are feasible and what features can be observed.

For a given environment θ, a trajectory ξ is generated by executing a policy under the transition function T_θ. Its features are represented by a function $\Phi(\xi, \theta)$, which depends on both the trajectory's state-action sequence and the environment. For example, the same path may traverse different terrain types depending on θ, resulting in different features. This makes the feature map $\Phi(\cdot, \theta)$ environment-dependent and highlights the role of environment design in shaping the information content of preference queries.

At each round i, the learner (robot):

1) Selects an environment $\theta^{(i)} \in \Theta$,
2) Generates a pair of trajectories $(\xi_A^{(i)}, \xi_B^{(i)})$ in $T_{\theta^{(i)}}$,
3) Queries the human: "Which trajectory do you prefer?",
4) Updates its belief over the reward weights $\mathbf{w}$.

This process allows the learner to actively shape the learning problem by selecting environments that yield the most informative trajectory comparisons for inferring the human's reward function.

IV. Technical Approach

Active preference learning faces challenges in generating trajectories that optimize for information gain (Eqn. 2),

Algorithm 1 Counterfactual Reasoning

Require: Belief $P(w)$, N samples, M subset size
1: Sample $\{w_1, \ldots, w_N\} \sim P(w)$
2: Select M diverse weights (e.g., max cosine distance)
3: **for** each selected w_k **do**
4: Generate trajectory ξ_k by maximizing $w_k^\top \Phi(\xi)$ // e.g., via RL or CHOMP
5: **end for**
6: Compute information gain f (Eq. 2) for all pairs ξ_i, ξ_j
7: Return most informative pair (ξ_i, ξ_j)

Algorithm 2 Environment Design

Require: Environment parameters Θ, Bayesian optimization iterations N
1: **for** $t = 1$ to N **do**
2: Propose θ^t using Bayesian optimization
3: Generate $(\xi_A^{(t)}, \xi_B^{(t)})$ via CR (Algorithm 1) in env θ^t
4: Compute information gain $F(\xi_A, \xi_B; \theta^t)$
5: Update GP model with $(\theta^t, F(\xi_A, \xi_B; \theta^t))$
6: **end for**
7: Return optimal θ^* found and corresponding (ξ_A, ξ_B)

as the objective function involves a pair of trajectories as variables. This task is further complicated by the fact that the optimization is typically constrained to a single environment, which may not adequately represent the full feature space, resulting in learned rewards that may fail to generalize effectively and suffer from sample inefficiency. Our approach using counterfactual reasoning and environment design to address these issues is summarized in Fig. 2.

A. Counterfactual Reasoning

Counterfactual reasoning explores different trajectories that could result if a hypothesized set of reward weights were the true weights. We maintain a belief over the weights (via an adaptive Metropolis algorithm [22]) while estimating the human's reward function, where each sample of weights could lead to a different policy and consequently different trajectories when the policy is executed. This allows us to pose counterfactual questions, such as "what if reward i is the true reward as opposed to reward j?" Let w_i be an instance of reward weights sampled from our belief. We can use reinforcement learning (RL) [20] to train a policy π_i that maximizes the reward function induced by w_i. Rolling out π_i in a given environment yields a trajectory ξ_i. Alternatively, we can directly optimize the trajectory with respect to $w_i^T \Phi(\xi)$ using trajectory optimization methods such as CHOMP [21] (Algorithm 1 lines 4).

By sampling reward weights and generating trajectories, we construct a set of counterfactual trajectories that represent different human preferences. We then evaluate the information gain objective f (Eq. (2)) for each pair of trajectories to identify the most informative preference query (Algorithm 1 lines 6-7). To minimize the number of evaluations of the objective function, we start by sampling N reward weights. We then select the most diverse M weights, where $M < N$, from this set by sequentially computing diversity based on cosine similarity, forming our final set of reward weights for evaluation (Algorithm 1 lines 1-2).

B. Environment Design

While counterfactual reasoning generates trajectory pairs optimizing for different reward weights, the fixed environment can limit their ability to reveal crucial preference distinctions. We posit that if we have the ability to "imagine" new environments, we can better generate trajectories that show the differences between the different reward weights.

To formalize this idea, we make explicit the role of environment parameters in trajectory generation. Recall that Θ denotes the set of configurable environment parameters where $\theta \in \Theta$ induces a transition function T_θ. The feature function Φ is environment-dependent and written as $\Phi(\xi, \theta)$. Let $F(\xi_A, \xi_B; \theta)$ denote the information gain from presenting the trajectory pair (ξ_A, ξ_B) in environment θ, defined analogously to Eqn. 2, but accounting for the dependence of trajectories and features on θ. We formulate environment design as a bilevel optimization problem [1]:

$$\max_{\theta \in \Theta} \max_{(\xi_A, \xi_B) \in \Xi(\theta)} F(\xi_A, \xi_B; \theta) \tag{3}$$

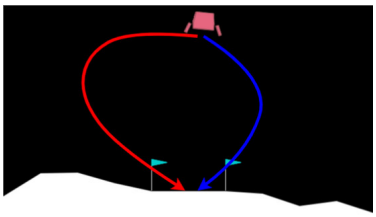
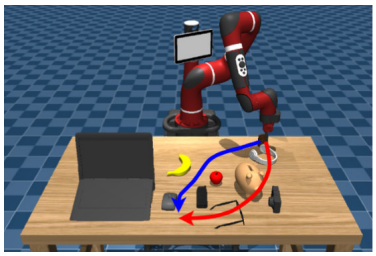
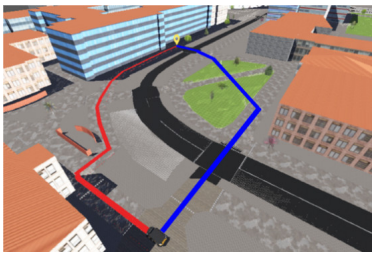
where $\Xi(\theta)$ denotes the set of feasible trajectory pairs under the MDP with transition T_θ. The outer optimization selects the environment parameters θ that maximize the informativeness of the resulting preference query, while the inner optimization identifies the most informative trajectory pair (ξ_A, ξ_B) that can be generated in that environment. This bilevel structure reflects the coupling between environment design and query synthesis in our framework.

Since $F(\xi_A, \xi_B; \theta)$ is generally not differentiable with respect to θ, we use Bayesian optimization [23], which models F with a Gaussian process (GP) defined by a mean function $m : \Theta \to \mathbb{R}$ and a positive definite covariance function $K : \Theta \times \Theta \to \mathbb{R}$. We use the upper confidence bound (UCB) acquisition function, selecting θ that maximizes $UCB(\theta) = \mu(\theta) + \kappa\sigma(\theta)$, where κ balances exploitation against exploration. We use the `BayesianOptimization` Python library [9] which employs a Matérn kernel with hyperparameters fit via maximum likelihood estimation. Unless otherwise specified, we use the library's default settings. Algorithm 2 shows the pseudocode for environment design.

V. Experiments

We evaluate CRED across a suite of simulation experiments and a user study. Through these experiments we demonstrate that CRED enables more accurate and efficient inference of human reward functions than existing methods and characterize the contributions of its core components.

[1]In principle, environment parameters θ and trajectory pairs (ξ_A, ξ_B) could be jointly optimized in a single-level formulation. However, the feasible set of trajectories $\Xi(\theta)$ depends on θ through the induced transition function T_θ, making joint optimization computationally expensive. The bilevel formulation leverages this structure by first selecting θ and then optimizing over $\Xi(\theta)$, which is more tractable in practice.

	Lunar Lander	Table Top	Navigation
Features Φ	vertical speed, horizontal position	objects that the end effector hovers over and the trajectory length	path length, terrain traversed (paved, grass, asphalt and concrete)
Env params θ	wind power, ranging from 1 to 20	object positions on the table	terrain type of road network
Preferences	approach direction, landing speed	objects to avoid hovering over	terrains to avoid and path length

Fig. 3: Examples of preference queries with details and visualization across the three simulation domains.

We begin with simulation experiments in three domains (Lunar Lander, Tabletop Manipulation, and Navigation) where we measure the accuracy of learned rewards and compare CRED against two established baselines [6], [2]. We conduct ablation studies to isolate the impact of counterfactual reasoning and environment design, followed by a comparison of our environment design strategy with domain randomization, an alternative that samples environments uniformly at random. Finally, we present user study results evaluating if real users can effectively teach preferences using CRED in manipulation and navigation tasks.

Our experiments address the following hypotheses:

- **H1:** CRED achieves a higher reward accuracy with fewer preference queries compared to baselines [6], [2].
- **H2:** Both counterfactual reasoning and environment design contribute significantly to CRED's performance
- **H3:** CRED outperforms domain randomization by generating more informative environments that accelerate reward learning.
- **H4:** Real users prefer interacting with CRED over baseline methods in practical manipulation and navigation tasks and report lower mental workload as measured by NASA-TLX.

A. Simulation Experiments

For all simulation experiments, we simulate 10 users, each initialized with a distinct ground-truth reward weight vector. To encourage diversity, we sample 1000 random weight vectors and select the cluster centers obtained via K-means as the ground-truth weights. This setup allows us to test whether CRED can generalize across heterogeneous user preferences.

1) Environment Setup: **Lunar Lander**. We use the Gymnasium implementation of Lunar Lander [24], where a lander must safely reach a designated pad in a 2D environment. Beyond the environment's default objective of safe landing, we introduce user-dependent preferences such as approaching from the left versus right side and controlling descent speed. The feature vector Φ encodes the lander's vertical velocity and horizontal position, while the environment parameter θ specifies the wind power applied during descent. To generate trajectories, we train PPO [25] agents using the Stable Baselines implementation. We use the default hyperparameters, with two exceptions: the target KL divergence is set to 0.01, and both the policy and value networks are two-layer MLPs with 256 hidden units per layer.

Tabletop Manipulation. We adapt a tabletop manipulation domain [26] in MuJoCo [27], where a robot delivers a cup of coffee across a cluttered table. To avoid spills on critical items such as electronics, the robot must learn user preferences for safe trajectories. Objects on the table include fruits, laptop, a phone, glasses, headphones, a camera, and a piggy bank. The features encode how often the robot's trajectory hovers above each object, while the environment parameters θ specify object placements. To make the search over possible object configurations tractable, we use a variational autoencoder (VAE) [28] to compress environments into a latent space Z (see Appendix VI-A), where the environment parameters θ correspond to latent vectors $z \in Z$. We use CHOMP [21] to generate trajectories and augment the cost function with $-R$ to incorporate the rewards.

Navigation. We consider a delivery task where a robot must transport food to a customer by navigating between predefined start and goal locations over a street network derived from OpenStreetMaps [29]. The network is simulated in Webots [30] for experiments and user studies. The features encode the path length and the proportion of each terrain type (e.g., asphalt, paved, grass, concrete, and brick) along the route. The environment parameters θ control the surface types assigned to the edges on the shortest path, thereby altering the trade-offs between efficiency and terrain preferences. Trajectories are generated using value iteration [20], where graph nodes correspond to states and outgoing edges define the available actions at each node.

2) Metrics: We measure accuracy using the sample correlation coefficient r between ground truth and estimated rewards:

$$r = \frac{\sum_{i=1}^{n} \left(R_{gt}^{i} - \bar{R}_{gt}\right)\left(R_{est}^{i} - \bar{R}_{est}\right)}{\sqrt{\sum_{i=1}^{n} \left(R_{gt}^{i} - \bar{R}_{gt}\right)^{2}} \sqrt{\sum_{i=1}^{n} \left(R_{est}^{i} - \bar{R}_{est}\right)^{2}}}, \tag{4}$$

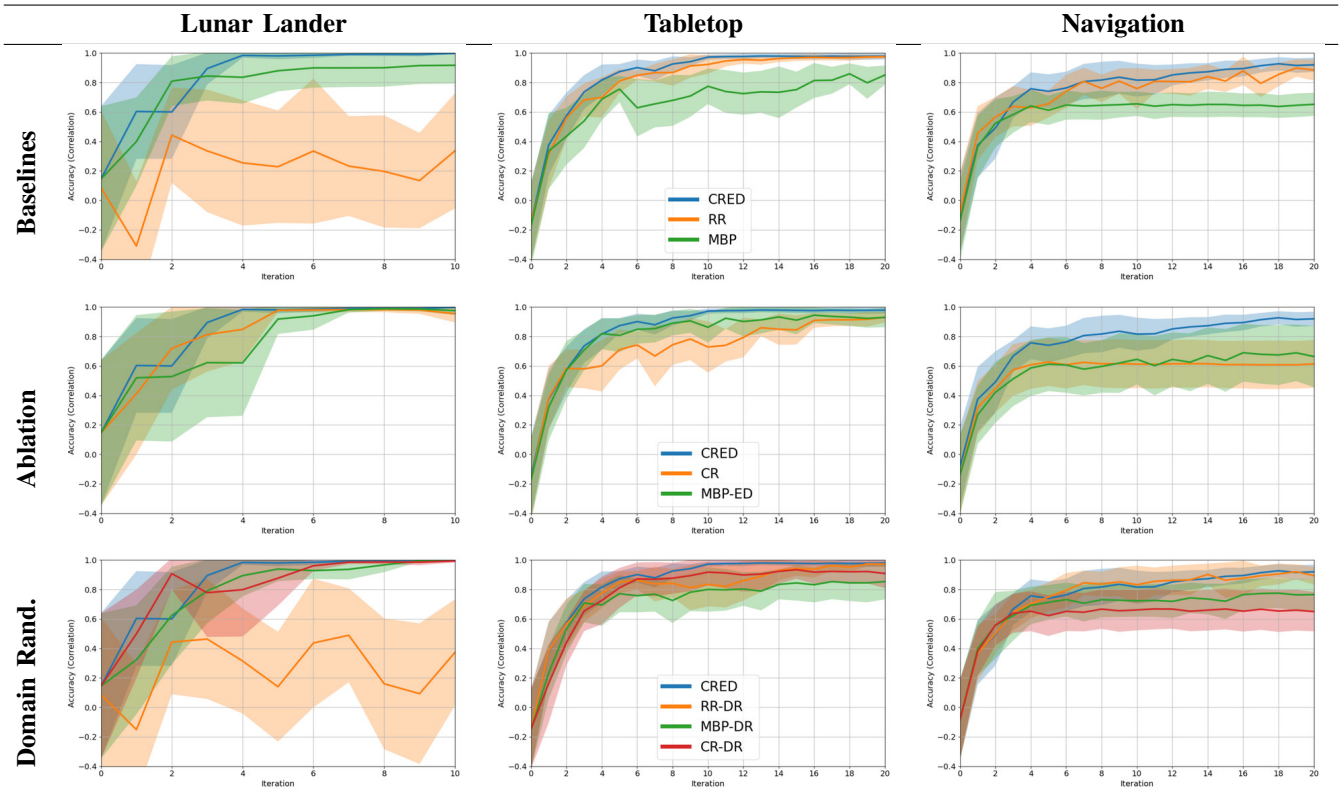


Fig. 4: Accuracy of the estimated rewards across three domains (columns) for each experiment (rows).

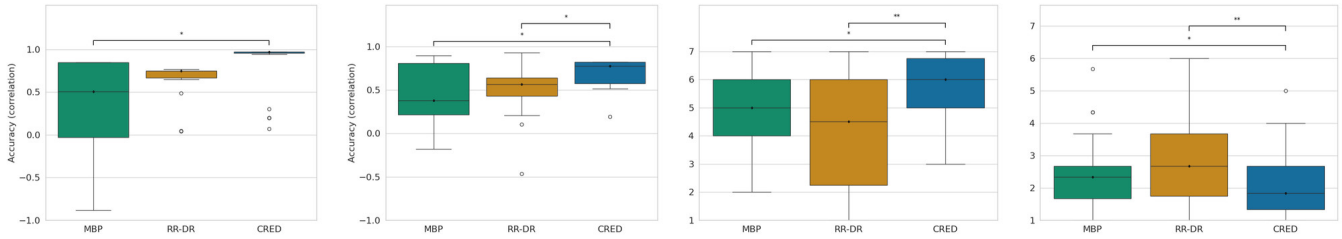
where R^i_{gt} and R^i_{est} are the ground truth and estimated rewards for sample i, and $\bar{R}$ denotes the sample mean. We compute R_{gt} and R_{est} by evaluating the true and learned weight vectors over a grid of feature vectors, with feature ranges determined from a set of trajectories sampled in each environment.

3) Baselines: We compare CRED against two state-of-the-art preference learning baselines. The first optimizes the mutual information objective (Eqn. 2) over pre-generated trajectories from random rollouts [6], which we denote as **RR**. The second, inspired by [2] and used in later work such as PEBBLE [7], learns a reward function by training a neural network on preference data. In our Bayesian belief framework, we replicate this baseline by using the mean belief as the reward function. During rollouts, the policy trained on the learned reward function follows its action with probability $1-\varepsilon$ and explores with probability ε ($\varepsilon = 0.25$ in our experiments). For CHOMP, exploration is simulated by adding Gaussian noise. We refer to this baseline as the **M**ean **B**elief **P**olicy (**MBP**).

4) Simulation Results: **Baseline Comparison**. Figure 4 (row 1) reports mean accuracy with 95% confidence intervals across all three domains, comparing CRED to RR [6] and MBP [2]. CRED consistently achieves the highest final accuracy, reaching 0.998, 0.979, and 0.906 in Lunar Lander, Tabletop, and Navigation, respectively. CRED also converges rapidly, surpassing 95% accuracy within 4 iterations for Lunar Lander and 10 for Tabletop. Navigation requires around 16 iterations, reflecting the greater difficulty of distinguishing preferences across multiple terrain types. By contrast, MBP plateaus around 60% accuracy and fails to produce informative queries after roughly 8 iterations. Because MBP cannot get feedback in novel environments, it is restricted to the features available in the current environment, limiting its effectiveness. Overall, CRED achieves both higher reward accuracy and faster convergence, supporting **H1**.

Ablations. To assess the contribution of each component in CRED, we evaluate two ablations. First, **CR** removes environment design, relying only on counterfactual reasoning. Second, **MBP-ED** removes counterfactual reasoning, applying environment design to the mean-belief policy baseline. As shown in row 2 of Figure 4, both ablations yield either lower final accuracy or slower convergence across all domains, indicating that environment design and counterfactual reasoning are complementary. These results support **H2**.

We further test the environment design strategy against **domain randomization (DR)**, which selects environment parameters $\theta \in \Theta$ uniformly at random. For fairness, the number of environments sampled under DR matches the number evaluated during Bayesian optimization. We construct three DR baselines: **RR-DR**, which generates random rollouts in randomized environments; **MBP-DR**, which executes trajectories from the mean-belief policy under random environments; and **CR-DR**, which applies counterfactual reasoning without guided environment selection. Results (row 3 of Figure 4) show that all DR variants underperform CRED (with the exception of RR-DR in Navigation), highlighting



(a) Tabletop accuracy (b) Navigation accuracy (c) Ease of choice (d) NASA-TLX

Fig. 5: Results from the user study: higher is better for (a)-(c), lower is better for (d). Within each plot the central line denotes the median, the upper and lower edges correspond to the first (Q1) and third (Q3) quartiles, and the whiskers extend to 1.5 times the interquartile range from Q1 and Q3. Statistically significant differences indicated as: * $p \leq 0.05$, ** $p \leq 0.01$.

the benefit of using Bayesian optimization to actively search for informative environments rather than sampling them uniformly at random (**H3**). We note that RR-DR does not enforce a goal-reaching constraint on trajectories, enabling greater diversity in the feature space. However, our user study (Sec. V-B) shows that RR-DR trajectories are harder for users to interpret, limiting their practicality despite their competitive accuracy in simulation.

B. User Study

1) Setup: We conducted an IRB-approved user study to evaluate CRED in both tabletop and navigation tasks. A total of 25 participants were recruited from the university campus (ages 20–39, $M = 26.8$, $SD = 4.5$; 18 male, 6 female, 1 other). Participants were randomly assigned to conditions in a within-subjects design, with each participant completing 6 preference queries per task. We compared CRED against two baselines: MBP [2] and RR-DR, which uses pre-generated trajectories [6] augmented with domain randomization. We did not include MBP with domain randomization (MBP-DR), as simulation results showed that it performed comparably to MBP alone. To avoid participant fatigue, we restricted the study to these three conditions. The order of conditions was randomized for each task to mitigate ordering effects.

In each trial, participants were presented with a pair of robot trajectories and asked to select the one they preferred. Trajectories were generated in simulation and visualized as arrows overlaid on the task environment, similar to Fig. 3. The reward functions inferred from these preferences were subsequently deployed on the physical robot platforms, as demonstrated in the supplementary video. For the tabletop task, the preferred behavior was to avoid passing over electronics, while for the navigation task, the preferred behavior was to avoid crossing asphalt roads. Participants were informed that although the robot could traverse grass, such routes may be less energy efficient. These task instructions were chosen to define a ground-truth reward function [2], allowing us to evaluate the accuracy of the learned rewards while also capturing realistic trade-offs in user preferences.

After each condition, participants completed a short questionnaire. Subjective workload was measured using the NASA-TLX [31] survey, and perceived ease of comparison was measured with a 7-point Likert item ("It was easy to choose between the trajectories that the robot showed me," 1=strongly disagree, 7=strongly agree). To assess learning performance, we compared the inferred reward functions to the ground truth rewards by computing the reward correlation (Eq. 4), which was also used in the simulation experiments.

2) Results: Fig. 5 shows the box plots of the user study results. For the tabletop task, CRED achieved higher reward correlation with the ground truth rewards (median $r = 0.97$, IQR $= 0.01$), outperforming RR-DR (median $r = 0.75$, IQR $= 0.08$) and MBP (median $r = 0.51$, IQR $= 0.88$). Similarly, in the navigation task, CRED again showed better performance (median $r = 0.78$, IQR $= 0.25$), compared to RR-DR (median $r = 0.57$, IQR $= 0.21$) and MBP (median $r = 0.38$, IQR $= 0.59$). In terms of subjective results, participants reported lower mental workload under CRED (NASA-TLX: median $= 1.83$, IQR $= 1.33$) compared to DR-RR $(2.67, 1.92)$ and MBP $(2.33, 1.00)$. They also rated CRED higher on ease of comparison $(6.0, 1.75)$ than DR-RR $(4.5, 3.75)$ and MBP $(5.0, 2.00)$, supporting **H4**.

Statistical comparisons were conducted using the Wilcoxon signed-rank test, a nonparametric test appropriate for ordinal measures (e.g., Likert ratings, NASA-TLX scores) and bounded non-normally distributed measures such as reward correlations. To account for multiple pairwise comparisons, we applied Holm–Bonferroni correction to control the family-wise error rate. CRED significantly outperformed both RR-DR and MBP in the tabletop and navigation tasks, with the exception of the CRED vs. RR-DR comparison in tabletop reward accuracy. We observed a few outliers in the tabletop task for both CRED and RR-DR, which may be attributable to participants not following task instructions. These results show that CRED more consistently recovers the true rewards across both navigation and manipulation domains. Furthermore, participants found CRED less mentally demanding and perceived the queries generated by CRED as easier to evaluate.

[2]Ground-truth rewards are defined as linear functions of trajectory features, normalized to unit-length weight vectors. For the tabletop task, features are the number of waypoints over `fruit`, `accessories`, and `electronics`, and the trajectory length, with weights $[-0.1, -0.1, -2.0, -1.0]$. For the navigation task, features are path length and the distances traversed on `paved`, `grass`, `asphalt`, and `concrete`, with weights $[-1.0, -0.1, -2.0, -5.0, -0.1]$.

VI. Conclusion

We introduce CRED, a query generation framework that jointly optimizes environments and trajectories for active preference learning. Experiments show that CRED achieves higher sample efficiency and reward accuracy than prior methods, and ablations highlight the importance of counterfactual reasoning, environment design, and Bayesian optimization for selecting informative environments. A user study in tabletop and navigation tasks demonstrates that CRED produces queries that are easier to answer and impose lower mental workload. CRED is applicable across policy classes (e.g., PPO, value iteration, trajectory optimization) and domains with different preferences. One limitation is the computational complexity: as a bilevel formulation, CRED requires an outer search over environment parameters and an inner search over trajectories. Future work can mitigate this cost through parallelization of environment evaluations and more efficient approximations of query informativeness.

APPENDIX

A. VAE for Tabletop Object Positions

We learn a low-dimensional latent space Z over tabletop object layouts with a convolutional VAE. Each environment is represented as a $C{\times}H{\times}W$ binary grid x (here $C{=}7$ object channels; $H{=}W{=}4$), where channel c encodes the occupancy of object c over the grid. The encoder $q_\phi(z\,|\,x)$ consists of two stride-2 convolutions (32 and 64 filters, kernel $3{\times}3$, ReLU), followed by flattening and linear heads to the mean $\mu(x) \in \mathbb{R}^d$ and log-variance $\log\sigma^2(x) \in \mathbb{R}^d$ of a diagonal Gaussian in Z (latent dimension $d{=}20$). We sample $z = \mu + \sigma \odot \varepsilon,\ \varepsilon \sim N(0,I)$. The decoder $p_\theta(x|z)$ maps z through a linear layer back to the flattened conv feature size, then uses two stride-2 transposed convolutions ($64{\to}32,\ 32{\to}C$; kernel $4{\times}4$, ReLU) with a final sigmoid to produce per-cell, per-object probabilities $\hat{x} \in [0,1]^{C\times H\times W}$.

References

[1] L. Mao, G. Warnell, P. Stone, and J. Biswas, "Pacer: Preference-conditioned all-terrain costmap generation," *IEEE Robotics and Automation Letters*, 2025.

[2] 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*, vol. 30, 2017.

[3] A. Zhang, H. Sikchi, A. Zhang, and J. Biswas, "Creste: Scalable mapless navigation with internet scale priors and counterfactual guidance," *arXiv preprint arXiv:2503.03921*, 2025.

[4] D. Sadigh, A. D. Dragan, S. S. Sastry, and S. A. Seshia, "Active preference-based learning of reward functions," in *Robotics: Science and Systems*, 2017.

[5] E. Biyik and D. Sadigh, "Batch active preference-based learning of reward functions," in *Conference on robot learning*. PMLR, 2018.

[6] E. Biyik, M. Palan, N. C. Landolfi, D. P. Losey, D. Sadigh, *et al.*, "Asking easy questions: A user-friendly approach to active reward learning," in *Conference on Robot Learning*. PMLR, 2020.

[7] K. Lee, L. M. Smith, and P. Abbeel, "Pebble: Feedback-efficient interactive reinforcement learning via relabeling experience and unsupervised pre-training," in *International Conference on Machine Learning*. PMLR, 2021, pp. 6152–6163.

[8] D. Ramachandran and E. Amir, "Bayesian inverse reinforcement learning." in *IJCAI*, vol. 7, 2007, pp. 2586–2591.

[9] F. Nogueira, "Bayesian Optimization: Open source constrained global optimization tool for Python," 2014. [Online]. Available: https://github.com/bayesian-optimization/BayesianOptimization

[10] N. Wilde, D. Kulić, and S. L. Smith, "Active preference learning using maximum regret," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020.

[11] H. Karnan, E. Yang, G. Warnell, J. Biswas, and P. Stone, "Wait, that feels familiar: Learning to extrapolate human preferences for preference-aligned path planning," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2024.

[12] R. Wang, J. Lehman, J. Clune, and K. O. Stanley, "Paired open-ended trailblazer (poet): Endlessly generating increasingly complex and diverse learning environments and their solutions," *arXiv preprint arXiv:1901.01753*, 2019.

[13] M. Dennis, N. Jaques, E. Vinitsky, A. Bayen, S. Russell, A. Critch, and S. Levine, "Emergent complexity and zero-shot transfer via unsupervised environment design," *Advances in neural information processing systems*, 2020.

[14] A. Kulkarni, S. Sreedharan, S. Keren, T. Chakraborti, D. E. Smith, and S. Kambhampati, "Designing environments conducive to interpretable robot behavior," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020, pp. 10 982–10 989.

[15] Y.-S. Tung, M. B. Luebbers, A. Roncone, and B. Hayes, "Workspace optimization techniques to improve prediction of human motion during human-robot collaboration," in *Proceedings of the 2024 ACM/IEEE International Conference on Human-Robot Interaction*, 2024.

[16] ——, "Improving human legibility in collaborative robot tasks through augmented reality and workspace preparation," in *6th International Workshop on Virtual, Augmented, and Mixed-Reality for Human-Robot Interactions (VAM-HRI)*. ACM, Mar 2023.

[17] Y. Zhang, M. C. Fontaine, V. Bhatt, S. Nikolaidis, and J. Li, "Multi-robot coordination and layout design for automated warehousing," in *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence, IJCAI-23*, E. Elkind, Ed. International Joint Conferences on Artificial Intelligence Organization, 8 2023.

[18] S. Bansal, R. Newbury, W. Chan, A. Cosgun, A. Allen, D. Kulić, T. Drummond, and C. Isbell, "Supportive actions for manipulation in human-robot coworker teams," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020.

[19] T. K. Buening, V. Villin, and C. Dimitrakakis, "Environment design for inverse reinforcement learning," in *Proceedings of the 41st International Conference on Machine Learning*, 2024, pp. 24 808–24 828.

[20] R. S. Sutton, "Reinforcement learning: An introduction," *A Bradford Book*, 2018.

[21] M. Zucker, N. Ratliff, A. D. Dragan, M. Pivtoraiko, M. Klingensmith, C. M. Dellin, J. A. Bagnell, and S. S. Srinivasa, "Chomp: Covariant hamiltonian optimization for motion planning," *The International journal of robotics research*, vol. 32, no. 9-10, pp. 1164–1193, 2013.

[22] H. Haario, E. Saksman, and J. Tamminen, "An adaptive Metropolis algorithm," *Bernoulli*, vol. 7, no. 2, pp. 223 – 242, 2001.

[23] J. Snoek, H. Larochelle, and R. P. Adams, "Practical bayesian optimization of machine learning algorithms," *Advances in neural information processing systems*, vol. 25, 2012.

[24] M. Towers, A. Kwiatkowski, J. Terry, J. U. Balis, G. De Cola, T. Deleu, M. Goulão, A. Kallinteris, M. Krimmel, A. KG, *et al.*, "Gymnasium: A standard interface for reinforcement learning environments," *arXiv preprint arXiv:2407.17032*, 2024.

[25] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[26] S. Peng, H. Chen, and K. Driggs-Campbell, "Towards uncertainty unification: A case study for preference learning," *arXiv preprint arXiv:2503.19317*, 2025.

[27] E. Todorov, "Mujoco: A physics engine for model-based control," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2012, pp. 5026–5033.

[28] L. P. Cinelli, M. A. Marins, E. A. B. Da Silva, and S. L. Netto, *Variational methods for machine learning with applications to deep networks*. Springer, 2021, vol. 15.

[29] OpenStreetMaps, "Planet dump retrieved from https://planet.osm.org," https://www.openstreetmap.org, 2017.

[30] Webots, "http://www.cyberbotics.com," open-source Mobile Robot Simulation Software. [Online]. Available: http://www.cyberbotics.com

[31] S. G. Hart and L. E. Staveland, "Development of nasa-tlx (task load index): Results of empirical and theoretical research," in *Advances in psychology*. Elsevier, 1988, vol. 52, pp. 139–183.