Adaptive Querying for Reward Learning from Human Feedback

Yashwanthi Anand, Nnamdi Nwagwu, Naomi T. Fitter and Sandhya Saisubramanian

***Abstract*— Learning from human feedback is a popular approach to train robots to adapt to user preferences and improve safety. Existing approaches typically consider a single querying (interaction) format when seeking human feedback and do not leverage multiple modes of user interaction with a robot. We examine how to learn a penalty function associated with unsafe behaviors, such as side effects, using *multiple* forms of human feedback, by optimizing the *query state* and *feedback format*. Our framework for *adaptive feedback selection* enables querying for feedback in critical states in the most informative format, while accounting for the cost and probability of receiving feedback in a certain format. We employ an iterative, two-phase approach which first selects critical states for querying, and then uses information gain to select a feedback format for querying across the sampled critical states. Our experiments demonstrate the sample efficiency of our approach. Experiment videos, code and appendices are found on our website: https://tinyurl.com/AFS-learning**

I. INTRODUCTION

Learning from human feedback is widely used to train robots, particularly when designing precise reward functions is difficult. Several prior works have examined various feedback modalities to improve robot performance and avoid undesirable consequences such as negative side effects (NSEs) [1]–[4]. An example of an NSE is a robot damaging items along the way when optimizing distance to transport an object to the goal (Fig. 1). The feedback may be provided in many forms, ranging from binary approval signals to action corrections, each varying in the granularity of information revealed to the robot and the human effort involved.

To efficiently balance the *trade-off* between seeking feedback in a format that accelerates robot learning and reducing human effort involved, it is beneficial to seek detailed feedback sparingly in certain states and complement it with feedback types that require less human effort in other states. That is, the robot queries in different formats in different regions. Such an approach could also reduce the sampling biases associated with learning from any one format, thereby improving learning performance [5]. In fact, a recent study indicates that users are generally willing to engage with the robot in more than one feedback format [6]. However, existing approaches rely on a single feedback format throughout the learning process and *do not support* gathering feedback in different formats in different regions of the state space [7].

We present a framework for *adaptive feedback selection* (AFS) that enables a robot to seek feedback in multiple formats during learning. How can a robot decide *when to query* and in *which format*, while accounting for feedback cost and likelihood of receiving a response? Our approach selects the query state and format, such that the overall information gain is maximized. In the interest of clarity, we discuss AFS in the context of minimizing NSEs but the framework is general and can be applied broadly.

All authors are with Oregon State University, Corvallis OR 97331, USA {anandy, nwagwun, fittern, sandhya.sai}@oregonstate.edu

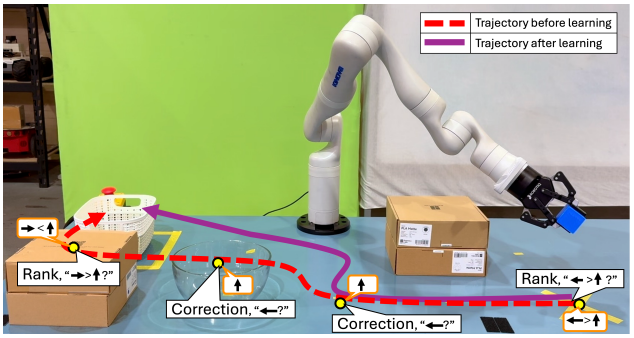


Fig. 1. An illustration of adaptive feedback selection. The robot arm learns to place the object in the basket without colliding with other objects in the way, by querying the human in different format across the state space.

The information gain of a feedback format is measured as the Kullback–Leibler (KL) divergence between the true NSE distribution–revealed to the robot via human feedback collected so far–and the robot's current knowledge of NSEs based on the feedback received so far. In each query cycle, the robot selects a feedback format that maximizes its information gain, given its current knowledge of NSEs.

When collecting feedback in every state is infeasible, the robot must prioritize querying in *critical states*—where the agent's learned model is most uncertain about NSE severity. This uncertainty is quantified using KL-divergence between the robot's current belief and human-provided feedback. In contrast, prior works select query states at random or along the shortest path to the goal, which may not result in a faithful NSE model [3], [4].

To efficiently gather information under a limited query budget, we use an iterative approach with the following key steps (outlined in Fig. 2(a)): (1) states are partitioned into clusters, with a cluster weight proportional to the number of NSEs discovered in it; (2) a critical states set is formed by sampling from each cluster based on its weight; (3) a feedback format that maximizes the information gain in critical states is identified, while accounting for the cost and likelihood of receiving a feedback, using the human feedback preference model; and (4) cluster weights and information gain are updated, and a new set of critical states are sampled to learn about NSEs, until the querying budget expires. The gathered information is mapped to a penalty function and augmented to the robot's model to compute an NSE-

minimizing policy for task completion.

Our key contributions are: (1) a novel framework for adaptive querying that selects both where and how to query for human feedback; (2) an algorithm for query selection that balances informativeness and human effort; and (3) empirical evaluation in simulation, using a user study and a Kinova Gen3 robot arm.

II. BACKGROUND

Markov Decision Processes (MDPs) are a popular framework to model sequential decision making problems and are defined by the tuple $M = \langle S, A, T, R, \gamma \rangle$, where S is the set of states, A is the set of actions, $T(s, a, s')$ is the probability of reaching state $s' \in S$ after taking an action $a \in A$ from a state $s \in S$ and $R(s, a)$ is the reward for taking action a in state s. An optimal deterministic policy $\pi^* : S \to A$, maximizes the expected reward. When the objective or reward function is incomplete, even an optimal policy can produce unsafe behaviors such as side effects.

Negative Side Effects (NSEs) are unintended, undesirable consequences of operating based on incomplete reward functions [3], [8], [9]. We mitigate NSEs by learning a penalty function from human feedback.

Learning from Human Feedback is a widely used technique to train robots when reward functions are unavailable or incomplete, including to improve safety [3], [4], [10], [11]. Feedback can take various forms such as *demonstrations* [12], [13], *corrections* [14], *critiques* [1], [3], and *ranking* trajectories [15]. Existing approaches focus on a single feedback type, limiting learning efficiency. Recent studies consider combinations such as demonstrations and preferences [16], [17], but assume a fixed order and do not scale to multiple formats. Another recent work examines feedback format selection by estimating the human's ability to provide feedback in a certain format [18]. In contrast, we dynamically select the most informative feedback *without* any pre-processing.

The information gain associated with a feedback quantifies how much the feedback improves the agent's understanding of the underlying reward function, often measured using Kullback-Leibler (KL) Divergence [18], $D_{KL}(P\|Q) = \sum_x P(x) \log \frac{P(x)}{Q(x)}$ where, P is the prior distribution and Q is the posterior distribution after observing evidence.

III. PROBLEM FORMULATION

Setting: Consider a robot operating in an environment modeled as a Markov Decision Process (MDP), using its acquired model $M = \langle S, A, T, R_T \rangle$. The robot optimizes the completion of its assigned task, which is its primary objective described by reward R_T. A *primary policy*, π^M, is an optimal policy for the robot's primary objective.

Assumption 1. Similar to [3], we assume that the robot's model M has all the necessary information to successfully complete its assigned task but lacks other superfluous details that are unrelated to the task.

Since the model is incomplete in ways unrelated to the primary objective, executing the primary policy produces negative side effects (NSEs) that are difficult to identify at design time. Following [3], we define NSEs as *immediate, undesired, unmodeled effects* of a robot's actions on the environment. We focus on settings where the robot has *no prior knowledge* about the NSEs of its actions. It learns to avoid NSEs by learning a penalty function $\hat{R}_N$ from human feedback, which is assumed to be consistent with true NSE penalty function R_N.

We target settings where the human can provide feedback in multiple ways and the robot can seek feedback in a *specific* format. This represents a significant shift from traditional active learning methods, which typically gather feedback only in a single format [3], [11]. The robot computes an NSE-minimizing policy to complete its task by optimizing: $R(s, a) = \theta_1 R_T(s, a) + \theta_2 \hat{R}_N(s, a)$, where θ_1 and θ_2 are fixed, tunable weights denoting priority over objectives.

Human's Feedback Preference Model: The user's *feedback preference model* is denoted by $D = \langle \mathcal{F}, \psi, C \rangle$ where,

- $\mathcal{F}$ is a predefined set of feedback formats the human can provide, such as demonstrations and corrections;
- $\psi : \mathcal{F} \to [0, 1]$ is the probability of receiving feedback in a format f, denoted as $\psi(f)$; and
- $C : \mathcal{F} \to \mathbb{R}$ is a cost function that assigns a cost to each feedback format f, representing the time or cognitive effort required from the human to provide that feedback.

The user's feedback preference model D may either be *pre-specified*, as in our simulations, or *learned* from user interactions prior to robot querying, as in our robot experiments. Abstracting user feedback preferences into probabilities and costs enables generalizing the preferences across similar tasks. We take the pragmatic stance that ψ is independent of time and state, denoting the user's preference about a format, such as not preferring formats that require constant supervision of robot performance. While this can be relaxed and the approach can be extended to account for state-dependent preferences, getting an accurate state-dependent ψ could be challenging in practice.

Assumption 2. Human feedback is immediate and accurate, when available.

A. Learning $\hat{R}_N$ from multiple forms of feedback

Since the agent has no prior knowledge about NSEs, it assumes none of its actions produce NSEs. We examine learning a penalty function $\hat{R}_N$ using the following popular feedback formats and their *annotated* (richer) versions. Each format provides varying levels of detail, leading to different learned reward models, as shown in Appendix A. In our settings, an action in a state may cause either mild, severe, or no NSEs. In practice, any number of NSE categories can be considered, provided the feedback formats align with them.

a) Approval: The robot randomly selects N state-action pairs from all possible actions in critical states and queries the human for approval or disapproval. Approved actions are labeled as acceptable, while disapproved actions are labeled as unacceptable. In *Annotated Approval*, the human also specifies the *NSE severity* (or category) for each disapproved action.

b) Corrections: The robot performs actions specified by its primary policy in the critical states, under human supervision. If the robot's action is unacceptable, the human intervenes with an acceptable action in these states. If all actions in a state lead to NSE, the human specifies an action with the least NSE. When interrupted, the robot assumes all actions except the correction are unacceptable in that state. In *Annotated Corrections*, the human also specifies the NSE severity when correcting an action.

c) Rank: The robot randomly selects N ranking queries of the form $\langle$*state*, *action 1*, *action 2*$\rangle$, by sampling two actions for each critical state. The human selects the safer action among the two options. The selected action is marked as acceptable and the other is treated as unacceptable. If both are safe or unsafe, one of them is selected at random.

d) Demo-Action Mismatch: The human demonstrates a safe action in each critical state, which the robot compares with its policy. All mismatched robot's actions are labeled as unacceptable. Matched actions are labeled as acceptable.

NSE Model Learning: We use l_m, l_h, and l_a to denote labels corresponding to mild, severe and no NSEs respectively. An acceptable action in a state is mapped to label l_a, and actions with mild NSEs are mapped to l_m. Actions that are considered to be unacceptable (without a specified severity) and those with high severity are mapped to l_h. Mapping feedback to these labels provides a consistent representation of NSE severity for learning under various feedback types. The NSE severity labels, derived from the gathered feedback, are generalized to unseen states by training a random forest classifier (RF) model to predict NSE severity of an action in a state. Any classifier can be used in practice. Hyperparameters for training are determined by a randomized search in the RF parameter space, using three-fold cross validation and selecting parameters with the least mean squared error for training and subsequently, determining the NSE severity. The label for each state-action pair is then mapped to its corresponding penalty value, yielding $\hat{R}_N(s,a)$. In our experiments, the penalties for l_a, l_m, and l_h are 0, -5, and -10 respectively.

IV. ADAPTIVE FEEDBACK SELECTION

Given an agent's decision making model M and the human's feedback preference model D, adaptive feedback selection (AFS) enables the agent to query for feedback in critical states in a format that maximizes its information gain.

Let p^* be the true underlying NSE distribution, unknown to the agent but known to the human. p^* deterministically maps state-action pairs to an NSE severity level (i.e., no NSE, mild NSE or severe NSE). Human feedback, when available, is sampled from this distribution. Let $p \sim p^*$ denote the distribution of state-action pairs causing NSE of varying severity, aggregated from all feedback received thus far. In other words, p represents the *accumulated* NSE information known to the agent, based on human feedback. Let q denote the agent's *learned NSE distribution*, based on all feedback received up to that point i.e., learned distribution from p.

AFS first selects critical states and then selects a feedback format for querying in these states, as outlined in Fig. 2(a).

A. Critical States Selection

Intuitively, when the budget for querying a human is limited, the robot must prioritize querying in *critical states*—states where the agent's NSE model diverges most from the true NSE distribution. We measure this learning gap using KL divergence between the distribution of data collected until iteration t (p^t), and the agent's learned NSE distribution from the previous iteration $t-1$ (q^{t-1}): $D_{KL}(p^t \| q^{t-1})$.

We compare p^t with q^{t-1} to identify states where the agent's NSE knowledge was incorrect, guiding the selection of the next batch of critical states. While $D_{KL}(p^t \| q^t)$ might seem like a better choice, it only measures how well the agent learned from feedback at t and does not highlight states where its prior NSE estimates were wrong. Alg. 1 outlines our approach for selecting critical states at each learning iteration, with the following three key steps.

1. Clustering states: Since NSEs are typically correlated with specific state features and do not occur at random, we partition the state space S into $\mathcal{K}$ clusters, grouping states with similar NSE severity. In our experiments, we use KMeans clustering algorithm with Jaccard distance to measure the distance between states based on their features. In practice, any clustering algorithm can be used, including manual clustering. The goal is to create meaningful partitions of the state space to select critical states for querying.

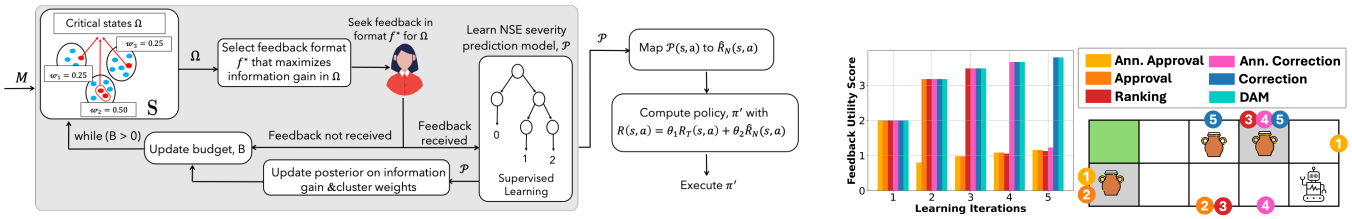
2. Estimating information gain: We define the information gain of sampling from a cluster $k \in K$, based on the learning gap discussed earlier,

$$IG(k)^t = \frac{1}{|\Omega_k^{t-1}|} \sum_{s \in \Omega_k^{t-1}} D_{KL}(p^t \| q^{t-1}) \tag{1}$$

$$= \frac{1}{|\Omega_k^{t-1}|} \sum_{s \in \Omega_k^{t-1}} \sum_{a \in A} p^t(a|s) \cdot \log\left(\frac{p^t(a|s)}{q^{t-1}(a|s)}\right) \tag{2}$$

where Ω_k^{t-1} denotes the set of states sampled for querying from cluster k at iteration $t-1$. $p^t(a|s)$ and $q^{t-1}(a|s)$ denote the NSE severity labels of action a in s, as provided in the feedback data and in the agent's learned model, respectively.

3. Sampling critical states: At each learning iteration t, the agent assigns a weight w_k to each cluster $k \in K$, proportional to the new information on NSEs revealed by the most informative feedback format identified at $t-1$, using Eqn. 2. Clusters are given equal weights when there is no prior feedback (Line 4). We sample critical states in batches but they can also be sampled sequentially. When sampling in batches of N states, the number of states n_k to be sampled from each cluster is determined by its assigned weight. At least one state is sampled from each cluster to ensure sufficient information for calculating the information gain for every cluster (Line 5). The agent randomly samples n_k states from corresponding cluster and adds them to a set of critical states Ω (Lines 6, 7). If the total number of critical states sampled is less than N due to rounding, then the



(a) (b)

Fig. 2. Solution approach overview. (a) The critical states Ω for querying are selected by clustering the states. A feedback format f^* that maximizes information gain is selected for querying the user across Ω. The NSE model is iteratively refined based on feedback. An updated policy is calculated using a penalty function $\hat{R}_N$, derived from the learned NSE model. (b) Feedback utility of different formats across iterations on a Vase domain instance. Circled numbers mark the identified critical states, with circle color indicating the feedback format.

Algorithm 1 Critical States Selection

Require: N, No. of critical states; $\mathcal{K}$, No. of clusters

1: $\Omega \leftarrow \emptyset$
2: Cluster states into $\mathcal{K}$ clusters, $K = \{k_1, \ldots, k_{\mathcal{K}}\}$
3: **for** each cluster $k \in K$ **do**
4: $\quad W_k \leftarrow \begin{cases} \frac{1}{\mathcal{K}}, & \text{if no feedback received in any iteration} \\ \frac{IG(k)}{\sum_{k \in K} IG(k)}, & \text{if feedback received} \end{cases}$
5: $\quad n_k \leftarrow \max(1, \lfloor W_k \cdot N \rfloor)$
6: $\quad$ Sample n_k states at random, $\Omega_k \leftarrow \text{Sample}(k, n_k)$
7: $\quad \Omega \leftarrow \Omega \cup \Omega_k$
8: $N_r \leftarrow N - |\Omega|$
9: **if** $N_r > 0$ **then**
10: $\quad k' \leftarrow \arg\max_{k \in K} W_k$
11: $\quad \Omega \leftarrow \Omega \cup \text{Sample}(k', N_r)$
12: **return** Set of selected critical states Ω

Algorithm 2 Feedback Selection for NSE Learning

Require: Querying budget B, User preference model D

1: $t \leftarrow 1$; $\mathcal{V}_f \leftarrow 0$ and $n_f \leftarrow 0$, $\forall f \in \mathcal{F}$
2: Initialize p and q assuming all actions are safe: $\forall a \in A, \forall s \in S : p(s,a) \leftarrow l_a$, $q(s,a) \leftarrow l_a$
3: **while** $B > 0$ **do**
4: $\quad$ Sample critical states Ω using Alg. 1
5: $\quad$ Query user in format f^*, selected using using Eqn. 4, across sampled Ω
6: $\quad$ **if** feedback received in format f^* **then**
7: $\quad\quad p^t \leftarrow$ Update distribution based on the feedback
8: $\quad\quad \mathcal{P} \leftarrow \text{TrainClassifier}(p^t)$
9: $\quad\quad q^t \leftarrow \{\mathcal{P}(s,a), \forall a \in A, \forall s \in S\}$
10: $\quad\quad$ Update $\mathcal{V}_{f^*}$ using Eqn. 3
11: $\quad\quad n_{f^*} \leftarrow n_{f^*} + 1$
12: $\quad B \leftarrow B - C(f^*)$; $t \leftarrow t + 1$
13: **return** NSE classifier model, $\mathcal{P}$

remaining N_r states are sampled from the cluster with the highest weight and added to Ω (Lines 9-12).

B. Feedback Format Selection

In each iteration, the robot evaluates each feedback format based on its likelihood of receiving a response, its informativeness (KL-divergence), and feedback cost. For example, if a user frequently provides corrections but rarely rankings, the algorithm adapts by prioritizing corrections in future queries.

The *information gain* of a feedback format f at iteration t is calculated using the KL divergence between p^t and q^t:

$$\mathcal{V}_f = \frac{1}{|\Omega|} \sum_{s \in \Omega} D_{KL}(p^t \| q^t) \mathbb{I}[f = f_H^t] + \mathcal{V}_f (1 - \mathbb{I}[f = f_H^t]). \tag{3}$$

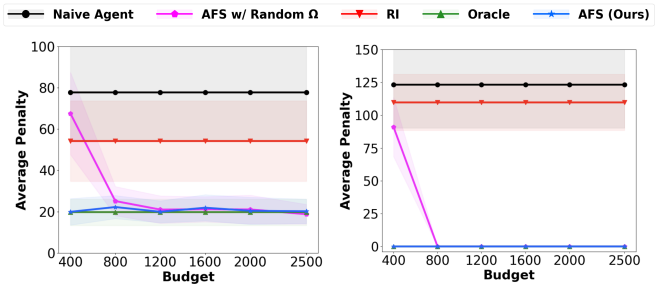
where, $\mathbb{I}[f = f_H^t]$ is an indicator function verifying whether the human's feedback format f_H^t, matches the requested format f. If no feedback is provided, the information gain of that format remains unchanged. A feedback format f^* is selected for querying using:

$$f^* = \underset{f \in \mathcal{F}}{\text{argmax}} \underbrace{\frac{\psi(f)}{\mathcal{V}_f \cdot \text{C}(f)} + \sqrt{\frac{\log t}{n_f + \epsilon}}}_{\text{Feedback utility of } f} \tag{4}$$

where $\psi(f)$ is the probability of receiving a feedback in format f and $C(f)$ is the feedback cost, determined using the human preference model D. t denotes the current learning iteration, n_f is the number of times f was received, and ϵ is a small value added for numeric stability.

Note that f^* is the *current* most informative feedback, based on the formats previously used. This may change as the agent explores and incorporates feedback of other formats. Thus, our approach effectively manages the trade-off between selecting feedback formats that were previously used to gather information and examining unexplored formats.

Alg. 2 outlines our feedback format selection approach. Since the agent has no prior knowledge of NSEs, all state-action pairs in p and q are initialized to safe action label (Line 2). This mapping will be refined based on the feedback received from the human, as learning progresses. It samples N critical states using Alg. 1 (Line 4). A feedback format f^* is selected using Eqn. 4. The agent queries the human for feedback in f^* (Line 5). The human provides feedback in format f^* with probability $\psi(f^*)$. Upon receiving the feedback, the agent updates the distribution, p^t based on the new NSE information, and trains an NSE prediction model, $\mathcal{P}$ (Lines 6-8). The agent's belief of NSE distribution q^t, is updated for all state-action pairs, using $\mathcal{P}$. The information gain $\mathcal{V}_{f^*}$ is updated using Eqn. 3, and n_{f^*} is incremented (Lines 9-11). This process repeats until the querying budget is exhausted, producing a predictive model of NSEs. Fig. 2(b)



(a) Vase (b) Push

Fig. 3. Average penalty incurred with different querying techniques.

illustrates the critical states and most informative feedback formats selected at each iteration in the Vase domain.

V. EXPERIMENTS

We conduct a number of simulated experiments in four domains: Vase, Push, Navigation, Atari Freeway[1] We also perform a user study where humans respond to agent queries in the Vase domain, and demonstrate our algorithm can be run on a real robot arm for a Drop-Object task.

Baselines (i) *Naive Agent*: The agent naively executes its primary policy without learning about NSEs, providing an upper bound on the NSE penalty incurred. (ii) *Oracle*: The agent has complete knowledge about R_T and R_N, providing a lower bound on the NSE penalty incurred. (iii) *Reward Inference with β Modeling (RI)* [18]: The agent selects a feedback format that maximizes information gain according to the human's inferred rationality β. (iv) *AFS w/ Random* Ω: The agent uses our AFS framework to learn about NSEs but with randomly sampled states for querying.

We use $\theta_1 = 1$ and $\theta_2 = 1$ for all our experiments. AFS uses learned $\hat{R}_N$. We simulate the feedback for a state-action pair using a softmax action selection [18], [19]. We optimize costs (negations of rewards) and compare techniques using NSE penalty and cost to goal, averaged over 100 trials.

Vase: The robot must quickly reach the goal, while minimizing breaking a vase as a side effect [20]. A state is represented as $\langle x, y, v, c\rangle$ where, (x, y) is the robot's coordinates. v indicates if a vase is present, and c indicates if the floor is carpeted. Breaking a vase placed on a carpet is a mild NSE and breaking a vase on the hard surface is a severe NSE. $\langle v, c\rangle$ are used for training. All instances have unavoidable NSEs.

Push: In this `safety-gymnasium` domain, the robot aims to push a box quickly to a goal state [21]. Pushing a box on a hazard zone produces NSEs. We modify the domain such that in addition to the existing actions, the agent can also *wrap* the box that costs $+1$. The NSEs can be avoided by pushing a wrapped box. A state is represented as $\langle x, y, b, w, h\rangle$ where, (x, y) are the robot's coordinates, b indicates carrying a box, w indicates if box is wrapped and h denotes if it is a hazard area. $\langle b, w, h\rangle$ are used for training.

[1]To focus more on the other experiments where we work with human users or a real robot, we present the results of the Navigation and Freeway simulations in Appendix A, which can be found on the https://tinyurl.com/AFS-learning.

TABLE I

AVG. COST AND STANDARD ERROR AT TASK COMPLETION.

Method	Vase: unavoidable NSE	Push: avoidable NSE
Oracle	54.46 ± 6.70	44.62 ± 9.97
Naive	36.00 ± 2.89	39.82 ± 5.44
RI	37.42 ± 1.01	42.15 ± 2.44
AFS (Ours)	63.00 ± 0.73	46.17 ± 0.86

A. Results in Simulation

There is a trade-off between optimizing task completion and mitigating NSEs, especially when NSEs are unavoidable. While some techniques are better at mitigating NSEs, they significantly impact task performance. Fig. 3 shows the average NSE penalty of different techniques, and Table I shows the average cost for task completion, averaged over 100 trials. *Lower* values are better for both NSEs and task completion cost. The baselines either mitigate NSEs or optimize cost, performing poorly on the other objective. The results show that our approach consistently mitigates avoidable and unavoidable NSEs, without affecting the task performance substantially. The results also show that AFS is more sample efficient, compared to baselines.

B. Calibration phase to learn user feedback preferences

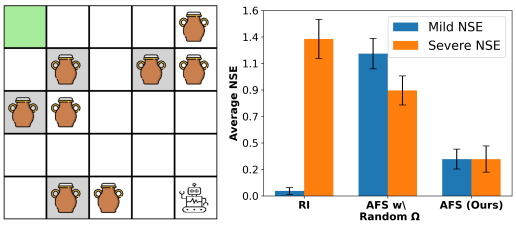
Our experiments with human subjects and the real robot experiment involve a calibration phase to *learn* the user feedback preference model, instead of simulating one.

User study: The users first complete a tutorial on different feedback formats described in Sec. III-A, after which we verify their ability to respond correctly. Each user is then prompted five times to provide feedback in each format, to which they can choose to respond or ignore the request, reflecting their interaction mode preference. The probability of receiving feedback in a given format is the fraction of prompts the user responded to, while the cost is based on the user's self-reported effort to provide the feedback.

Robot experiment: The calibration phase for the robot experiment is similar as above, but with four feedback formats: Correction, Approval, Ranking and Demo-Action Mismatch. Corrections and Demo-Action Mismatch involve physically moving the arm to a safe end effector state, and Approval and Ranking use keyboard inputs.

C. Simulated Vase Domain User Study

We conducted an IRB-approved within-subjects pilot study on a 5×5 Vase domain shown in Fig. 4(a), with 12 human participants, instead of simulated experts. Each user's feedback preference model was learned in the calibration phase. The study evaluates (1) AFS' effectiveness in minimizing NSEs by querying based on the learned feedback preferences; and (2) whether the selected feedback formats and critical states enhance agent's learning and align with user preferences. A detailed description of the study setup is in Appendix B.

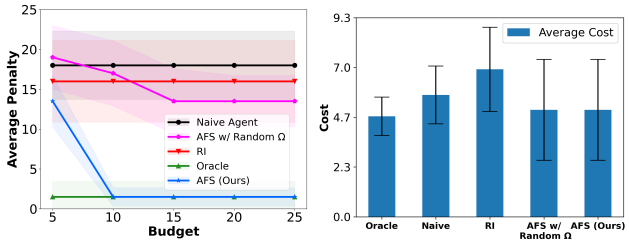


(a) Vase domain setup (b) Average # NSE

Fig. 4. User study results.

TABLE II
USERS' QUALITATIVE ASSESSMENT

Approach	Intelligent Feedback	Critical Points (%)			Improved Performance (%)	
		Yes	No	Overlap	Yes	No
RI	3.33 ± 1.23	83.30± 0.37	16.70± 0.37	73.47± 5.49	91.70± 0.28	8.30 ± 0.28
AFS w/ Random Ω	2.82 ± 0.94	66.70± 0.47	33.30± 0.94	75.51± 5.27	41.70± 0.49	58.30± 0.49
AFS (Ours)	3.25 ± 0.83	100.00± 0.00	0.00 ± 0.00	81.63± 4.94	100.00± 0.00	0.00 ± 0.00



(a) Average penalty (b) Average cost at task completion

Fig. 5. Results on Kinova Arm

Fig. 4(b) shows that our approach results in fewer NSEs, compared to the baselines. Since the NSE penalty is an aggregate measure that obscures severity distribution, we report exact NSE encounters by category for the user study. Table II reports average over responses to our questions: "On a scale of 1 to 5, how intelligent do you think the agent's choice of feedback formats are, given your preferences?", "Were the states in which the agent requested for feedback critical to its learning?", overlap between user-identified important query points and query points chosen by each approach, and "Did the agent's performance improve at the end of the learning phase?". Overall, our user study results indicate that our approach effectively selects query points and leads to improved learning outcomes.

D. Evaluation using Kinova Gen3 7DOF Arm

We evaluate our approach using a Drop-Object task (setup in Fig. 1). The robot's state space is discretized and denoted by $\langle x, y, o, f \rangle$ with end-effector coordinates (x, y), binary variables o and f denoting the presence of an obstacle and its fragility, respectively. Collisions with fragile obstacles are severe NSEs. Collisions with non-fragile objects produce mild NSEs. The NSE predictive model is trained on $\langle o, f \rangle$. Fig. 5 shows the average penalty incurred over 10 trials, with different techniques and using a learned preference model. Our approach incurs the least NSE penalty.

VI. SUMMARY AND FUTURE WORK

We present an information-theoretic approach to learn a reward function by strategically selecting critical query states and the most informative feedback format, considering the cost and uncertainty of each format. Our evaluations in simulation, user study and on a robot arm demonstrate the sample efficiency of our approach in learning to mitigate NSEs. In the future, we aim to extend our approach to continuous settings and add visualization to ensure that the queries are easily answerable.

REFERENCES

[1] Y. Cui and S. Niekum, "Active reward learning from critiques," in *ICRA*, 2018.
[2] D. Hadfield-Menell, S. J. Russell, P. Abbeel, and A. Dragan, "Cooperative inverse reinforcement learning," *NeurIPS*, vol. 29, 2016.
[3] S. Saisubramanian, E. Kamar, and S. Zilberstein, "A multi-objective approach to mitigate negative side effects," in *IJCAI*, 2021.
[4] S. Zhang, E. Durfee, and S. Singh, "Querying to find a safe policy under uncertain safety constraints in markov decision processes," in *AAAI*, 2020.
[5] S. Saisubramanian, E. Kamar, and S. Zilberstein, "Avoiding negative side effects of autonomous systems in the open world," *Journal of Artificial Intelligence Research (JAIR)*, vol. 74, 2022.
[6] S. Saisubramanian, S. C. Roberts, and S. Zilberstein, "Understanding user attitudes towards negative side effects of AI systems," in *Conference on Human Factors in Computing Systems*, 2021.
[7] 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 *IJCAI*, 2021.
[8] V. Krakovna, L. Orseau, M. Martic, and S. Legg, "Measuring and avoiding side effects using relative reachability," *arXiv preprint arXiv:1806.01186*, 2018.
[9] A. Srivastava, S. Saisubramanian, P. Paruchuri, A. Kumar, and S. Zilberstein, "Planning and learning for non-markovian negative side effects using finite state controllers," in *AAAI*, 2023.
[10] D. Hadfield-Menell, S. Milli, P. Abbeel, S. J. Russell, and A. Dragan, "Inverse reward design," *NeurIPS*, vol. 30, 2017.
[11] R. Ramakrishnan, E. Kamar, D. Dey, E. Horvitz, and J. Shah, "Blind spot detection for safe sim-to-real transfer," *JAIR*, vol. 67, 2020.
[12] D. Ramachandran and E. Amir, "Bayesian inverse reinforcement learning," in *IJCAI*, 2007.
[13] D. Brown and S. Niekum, "Efficient probabilistic performance bounds for inverse reinforcement learning," in *AAAI*, 2018.
[14] A. Bobu, M. Wiggert, C. Tomlin, and A. D. Dragan, "Feature expansive reward learning: Rethinking human input," in *HRI*, 2021.
[15] D. Brown, R. Coleman, R. Srinivasan, and S. Niekum, "Safe imitation learning via fast bayesian reward inference from preferences," in *ICML*, 2020.
[16] 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 (IJRR)*, 2022.
[17] B. Ibarz, J. Leike, T. Pohlen, G. Irving, S. Legg, and D. Amodei, "Reward learning from human preferences and demonstrations in atari," *NeurIPS*, vol. 31, 2018.
[18] G. R. Ghosal, M. Zurek, D. S. Brown, and A. D. Dragan, "The effect of modeling human rationality level on learning rewards from multiple feedback types," in *AAAI*, 2023.
[19] H. J. Jeon, S. Milli, and A. Dragan, "Reward-rational (implicit) choice: A unifying formalism for reward learning," *NeurIPS*, vol. 33, 2020.
[20] V. Krakovna, L. Orseau, R. Ngo, M. Martic, and S. Legg, "Avoiding side effects by considering future tasks," in *NeurIPS*, vol. 33, 2020.
[21] J. Ji, B. Zhang, J. Zhou, X. Pan, W. Huang, R. Sun, Y. Geng, Y. Zhong, J. Dai, and Y. Yang, "Safety gymnasium: A unified safe reinforcement learning benchmark," in *NeurIPS*, 2023.