Active Reward Learning from Online Preferences

Vivek Myers[1], Erdem Bıyık[2], Dorsa Sadigh[1,2]

Abstract— **Robot policies need to adapt to human preferences and/or new environments. Human experts may have the domain knowledge required to help robots achieve this adaptation. However, existing works often require costly offline re-training on human feedback, and those feedback usually need to be frequent and too complex for the humans to reliably provide. To avoid placing undue burden on human experts and allow quick adaptation in critical real-world situations, we propose designing and sparingly presenting easy-to-answer pairwise action preference queries in an online fashion. Our approach designs queries and determines when to present them to maximize the expected value derived from the queries' information. We demonstrate our approach with experiments in simulation, human user studies, and real robot experiments. In these settings, our approach outperforms baseline techniques while presenting fewer queries to human experts. Experiment videos, code and appendices are found on our website: `http://tinyurl.com/online-active`**

I. Introduction

Using trained robot policies in a zero-shot manner in real-world scenarios is challenging for many reasons, including insufficient training data, changing preferences of human users, or uncertainties present in dynamic environments. Today's robot algorithms need to fine-tune and adapt to the specifics of a given environment or user preferences in an online fashion. For example, the robot may need to safely explore the new environment and update its policy, but this is often too costly. An alternative is to require the human intervention by providing more data (such as demonstrations, physical feedback, language corrections) [1], or by formally specifying new reward functions [2], [3], which might be too difficult and expensive. In addition, the common paradigm of learning from such feedback requires the robot to run its old policy, gather information during this run, pause to train on this data, update the policy based on the newly provided data, and to only then initiate a new run. This is highly impractical in online settings, as we already observe in interactive imitation learning [4]–[8].

Instead, we would like the robot to reduce its uncertainty and update its policy by effectively asking humans questions and learning from their responses in an *online fashion*. This requires the robot to generate informative online questions that i) reduce the robot's uncertainty, ii) can easily be answered by humans, and iii) are critically timed, i.e., the robot should query humans infrequently and at the right time. Prior work has discovered pairwise comparisons as an effective form of feedback that can easily be answered by humans to update the robot's policy [9]–[14]. For instance, active preference-based learning techniques are data-efficient approaches that query humans with the most informative pairwise comparison—asking the human to compare two different robot trajectories—based on optimizing an information theoretic objective such as mutual information [15], volume removal [16], [17], maximum regret [18], [19], dissimilarity [20], [21], or determinantal point processes [22]. However, these works still require the policy to be updated during an offline training phase making them impractical for online policy updates. The online setting we are interested in best captures many real-world scenarios in which a robot is deployed in a high-value situation where it is worth querying humans during the run to avoid poor performance. These scenarios include self-driving settings, where bothering a human is worth it to avoid an accident; high-level task planning settings where robots need efficient clarification about which task to perform next; and manipulation settings such as pushing, grasping, door opening, etc. where the human may have strong preferences about the precise task to perform. In such real-world settings, we additionally wish to avoid asking too many questions so that humans can quickly and reliably respond to questions when they are most needed.

This brings us to our key insight: we would like to model the robot's epistemic uncertainty – its uncertainty about the environment – captured by the robot's reward function. This uncertainty allows the robot to evaluate *when* to ask *the most informative* pairwise comparisons to the human. Specifically, we take a multitask learning perspective to this problem and pretrain a library of robot reward functions on a number of tasks. In test time, we maintain a posterior over different tasks by directly modeling the effect of presenting queries to a human on the robot's belief state. We can thus compute the expected value of information (EVOI) [23], [24] of any potential pairwise query corresponding to the expected value the robot gains by asking that question. Using the EVOI metric, we propose an approach for selecting when to query a human expert and what queries to make. This is demonstrated through an example in Figure 1: here, the robot uses our approach to query a human expert and push an object to a preferred goal location. When the expected value of a question is high enough, we query the pair of actions with the highest expected value of information. Our approach asks pairwise comparison queries over the robot's actions (or the next short sequence of actions) that can be conveniently answered by humans. In addition, leveraging the EVOI metric allows us to ask the most informative questions at the critical time steps and thus update our policy

The authors would like to acknowledge the funding provided by the DARPA YFA award; NSF Awards #2218760, #1953032, and #1941722; and the Office of Naval Research.

[1]Computer Science, Stanford University, [2]Electrical Engineering, Stanford University. Emails: `vmyers`, `ebiyik`, `dorsa@cs.stanford.edu`

in an online fashion.

In sum, we make the following contributions:

1) We propose a formulation of the online pairwise action preference querying problem using a multitask setting to model reward uncertainty, and show how to use this formulation to compute the value of asking questions.
2) We show that our approach outperforms baselines by attaining a greater score while asking fewer questions in a tabular GridWorld game, a driving simulation, and continuous-action-space object-pushing robot environment.
3) We conduct a user study with the driving simulation, showing users prefer the questions and the performance of our approach compared to baselines. We further demonstrate that our approach transfers to a real robot task allowing it to move an object to a preferred location.

II. Related Work

Existing work on incorporating human feedback in robot learning has focused on interactive imitation learning as well as work on learning from other sources of data such as physical feedback, language instructions and corrections, rankings or pairwise comparisons. These works generally require extensive human feedback and/or learn from experts offline, whereas our approach is the only, to our knowledge, to both learn from humans using simple pairwise queries and perform that querying in an online fashion.

Utilizing Human Experts Online. Most prior work on adaptive online querying of human experts has focused on requesting full demonstrations. DAgger requires rollouts from a policy that uses expert predictions [4], while ThriftyDAgger [8] temporarily shifts control to human experts for partial demonstrations in risky or uncertain situations. Similarly, approaches such as uncertainty-aware action advising request demonstrations in high-uncertainty situations [25]. Building upon these interactive imitation learning work, HG-DAgger [5] and EIL [26] require human experts to decide when to stage interventions. These works require access to expert demonstrations which can be costly for robots with high degrees of freedom [27]–[31], and further require a training phase based on the collected data.

To avoid excessively burdening humans experts by requiring full demonstrations, Cohn *et al.* [24] leveraged EVOI and assumed the robot can move to any state to ask for the optimal action in that state. Approaches such as TAMER, COACH, and variants request continuous scalar feedback from human experts [32]–[35]. Other works tap into other sources of data such as physical corrections, language instructions or corrections, images, and rankings [1], [14], [36], [37]. While these works provide effective interfaces for collecting human data, they all require an offline training phase which does not allow for seamless online integration of new user inputs.

Preference-based Learning. Many works have examined reinforcement learning in the context of human preferences [9]. In particular, past work tackled reward learning in the context of preference-based queries [10], [13], [15]–[17], [38]–[41]. These works adaptively request human rankings or comparisons over trajectories to infer reward functions. Habibian *et al.* [42] expand these techniques to ask questions that demonstrate to human experts that the robot is learning from their instruction, while Biyik *et al.* [15] and Jeon *et al.* [43] propose supplementing preferences with diverse forms of human feedback. Meanwhile, within the student-teacher reinforcement learning framework, Zimmer *et al.* [44] propose an approach where a teacher agent gives action preferences to a student agent to help it learn quickly.

Other approaches such as T-REX attempt to learn reward functions that allow effective policies to be trained from rankings of suboptimal trajectories [45], while D-REX builds on this approach to learn from suboptimal trajectories without requiring expert rankings [46].

Outside of reinforcement learning, preference-based active learning has been used for classification tasks [47], [48] and ranking aggregation [47]. None of these approaches have tackled our problem of adaptively asking for pairwise comparisons between actions at deployment to conduct reward learning in an online setting.

III. Learning When to Make Informative Queries

We represent the robot's uncertainty as a distribution over possible reward functions. This distribution represents variability in human preferences about how a task should be performed. We assume our reward function in our environment is parameterized by a hidden latent task vector ω. The robot does not have access to this hidden task vector during training or evaluation. However, it may during evaluation at any point ask the human for their preference between two actions. The human responds with their preference (up to some noise factor), assuming access to the true task vector.

Formalization. We consider a Markov Decision Process (MDP) of the form (S, A, P, R^{ω}) where ω is a task representation parameterizing the reward function R^{ω}. We assume there is some distribution $\omega \sim \Omega$ which is known at train time and unknown at evaluation time. Having access to ω at train-time represents that we are able to learn different goal-conditioned policies during training, but must rely on human feedback to get information about the goal when deployed during evaluation.

Let us define $Q^*(s, a; \omega)$ to be the optimal Q-function for the MDP (S, A, P, R^{ω}) with optimal policy $\pi^*(s) = \arg\max_a Q^*(s, a)$. At evaluation-time, at a given state $s_t \in S$ at time t, we allow the robot to make an instantaneous action query of the form $(a_t^1, a_t^2) \in A \times A$, asking an expert to choose between the two possible actions. Denote the state the agent is currently in as $s \in S$. We assume the human responses follow the Boltzmann-rational response model:

$$H^{\pi}(a_1, a_2, s, \omega) = \frac{1}{1 + e^{\beta(Q^{\pi}(s, a_2; \omega) - Q^{\pi}(s, a_1; \omega))}} \quad (1)$$

with a fixed precision constant β (higher values indicate a more accurate human response model). The human returns a_1 with probability $H^{\pi^*}(a_1, a_2, s, \omega)$ and returns a_2 otherwise with probability $H^{\pi^*}(a_2, a_1, s, \omega)$. Assuming this response

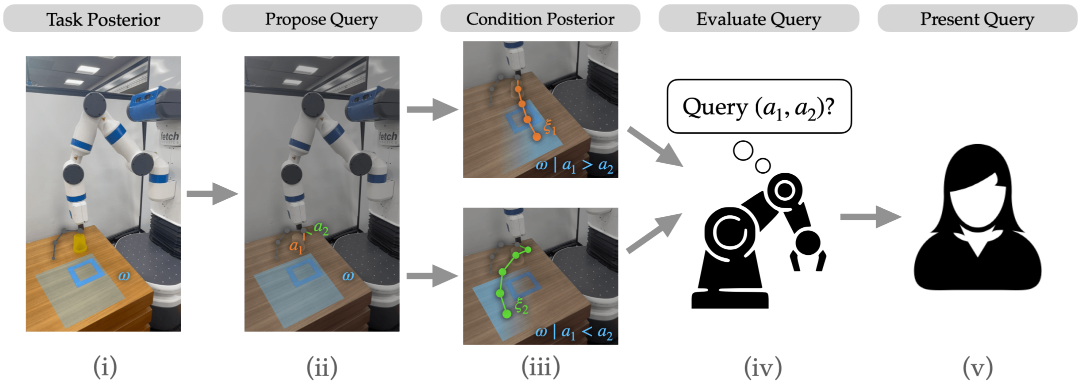

Fig. 1: Overview of our approach to decide if and what to query each time step. The robot must move the yellow cup to the goal location desired by the human, denoted by the blue tape square. The robot does not observe this goal location, and must query the human to determine where to move the cup. (i) Keep a posterior ω over the task (location of the dark blue tape square) conditioned on all past queries made to the human during the trajectory. (ii) Consider potential preference queries (a_1, a_2) between two actions. (iii) Condition the posterior on each possible response to the query. (iv) Use human model along with expected return of trajectories ξ_1 and ξ_2 under conditioned ω posteriors to compute value of query. Repeat for several query pairs of actions. (v) If the computed value of any potential query exceeds threshold, send the highest-value query considered to the human.

model for pairwise comparisons between action queries, our goal is to find out *when* to ask *what* questions to the user. This translates to the objective of asking informative questions that *maximize the reward* R^ω at evaluation, while also *minimizing the number of queries to the human expert*. We next formalize this joint optimization.

A. Querying Action Preferences

Training. During training, we assume we have access to a distribution of training tasks Ω, or samples from this distribution $\omega_1, \ldots, \omega_n \sim \Omega$. We train policies across the distribution Ω with a method that learns a Q-function, such as DQN [49]. We thus obtain Q functions that are conditioned on the hidden task vector as $Q^\pi(s, a; \omega)$.

Querying. We propose an approach based on the expected value of information (EVOI) [23] for determining when it is optimal to make queries. Our key insight is that using a family of task-parameterized Q-functions, we can approximate the expected value of asking a question to an expert. We then use a threshold c over the computed EVOI to determine when to generate a query. Intuitively, the threshold c determines the willingness of the robot asking a question.

At evaluation, we maintain a posterior over the task vector ω. Using this posterior, we can decide on which actions to take as well as compute the EVOI of asking a human expert for a preference between two actions of the form (a_1, a_2). We initialize the posterior Ω using the prior from the environment if it is known, and otherwise model the prior as a uniform distribution over the sampled tasks we are given. We update this posterior in response to each successive query made to the human expert, using the human response model. Writing the past set of responses, query states, and queries as $\mathcal{D} = \Big((a^{(1)}_{i_1}, s^{(1)}, (a^{(1)}_1, a^{(1)}_2)), (a^{(2)}_{i_2}, s^{(2)}, (a^{(2)}_1, a^{(2)}_2)), \ldots, (a^{(n)}_{i_n}, s^{(n)}, (a^{(n)}_1, a^{(n)}_2))\Big)$ for each $i_\bullet \in \{1, 2\}$, we obtain the following posterior update formula

$$\Pr[\omega \mid \Omega, \mathcal{D}] \propto \Pr[\omega \mid \Omega] \prod_{j=1}^{n} H(a^{(j)}_{i_j}, a^{(j)}_{3-i_j}, s^{(j)}, \omega). \tag{2}$$

We marginalize across this posterior at evaluation to obtain the following evaluation policy $\pi(s) = \arg\max_a \mathbb{E}_{\omega|\Omega,\mathcal{D}} Q^\pi(s, a; \omega)$. To compute the EVOI, we calculate the expected improvement in the expected Q value of the action taken by the robot, marginalizing across the ω posterior over the human goal belief conditioned on past query responses. The formulation selects queries based on how much they increase the value function evaluated on the current state, which we can model using our human response model and the effect of the queries on our posterior between reward functions.

One key assumption we make here in defining the EVOI is that the optimal policy given our belief over ω is well-represented by Q-values $\mathbb{E}_{\omega|\Omega,\mathcal{D}} Q^\pi(s, a; \omega)$. We make this assumption to avoid the computational burden of needing to refit policies for every possible distribution $\Omega \mid \mathcal{D}$ at each time step. This heuristic of setting the expected Q value to the weighted Q value of policies parameterized across tasks does result in optimistic myopic errors when optimal task policies will disagree substantially about which action to take in future.

A second assumption is made in the pessimistically myopic nature of this EVOI objective—the objective approximates the expected gain in value function at the current state when a query is asked assuming no further queries will be made during the current trajectory.

Intuitively, these two forms of myopic error may actually counteract each other. In cases where the Q function approximation is overconfident due to the first assumption (the optimal task policies will disagree in the future), the EVOI for querying will likely be high anyways at those future points of disagreement. This results in the Q value heuristic overestimate capturing the fact that the approach will do better than the myopic optimal policy that makes no more queries. In other words, the heuristic of using the expected Q across tasks is sometimes overly optimistic, but in such cases the overconfidence stems from future states with high EVOI where uncertainty will be resolved anyways. In practice, we indeed find that this heuristic of using expected Q performs well in our experimental settings.

Denote $\mathcal{D}_1 = \mathcal{D} \cup \{(a_1, s, (a_1, a_2))\}$ and $\mathcal{D}_2 = \mathcal{D} \cup \{(a_2, s, (a_1, a_2))\}$. We obtain the following formula for the EVOI of the query (a_1, a_2). A complete derivation of this formula can be found in Appendix A.

$$\begin{aligned}
&\mathrm{EVOI}(a_1, a_2) = \\
&\quad \mathop{\mathbb{E}}_{\omega|\Omega,\mathcal{D}} \Big[H^\pi(a_1, a_2, s, \omega) \mathop{\mathbb{E}}_{\omega'|\Omega,\mathcal{D}_1} \max_a \left[Q^\pi(s, a; \omega')\right] \\
&\qquad + H^\pi(a_2, a_1, s, \omega) \mathop{\mathbb{E}}_{\omega'|\Omega,\mathcal{D}_2} \max_a \left[Q^\pi(s, a; \omega')\right] \\
&\qquad - \max_a Q^\pi(s, a; \omega) \Big] \qquad (3)
\end{aligned}$$

When the EVOI of any query at the current state exceeds the query threshold c we make the query with the highest EVOI and update the posterior appropriately.

B. Continuous Action Spaces

In settings with a continuous action space, two issues arise: (1) computing Equation (3) for every pair of actions and (2) computing the $\max_a Q$ terms in Equation (3) for any action becomes intractable.

To solve (1), we randomly sample a fixed number of potential queries from the action space, and query the highest-EVOI of these queries if it exceeds c. To solve (2), we assume access to task-specific policies $\pi(s; \omega) = \arg\max_a Q^\pi(s, a; \omega)$, allowing us to compute $\max_a Q^\pi(s, a; \omega) = Q^\pi(s, \pi(s; \omega); \omega)$. Actor-critic approaches can be used to maintain an approximate $\pi(s; \omega)$ value [50], [51], allowing us to approximate Equation (3). In the continuous setting when not querying, we approximate the optimal action by using the policy $\pi(s) = \mathbb{E}_{\omega|\Omega,\mathcal{D}}\, \pi(s; \omega)$.

Selecting Hyperparameters. To select the constant β, we pick a value consistent with the scale of the rewards in our environment that models our belief over the accuracy of the human experts in intuitively assessing the goal-conditioned optimal Q-functions. To select c, we find a value that empirically yields the desired number of queries made by our agent either in simulation or by examining data from another source such as the replay buffer during training.

IV. Experiments

We conduct a number of simulated experiments in a GridWorld[1] and a driving environment, perform a user study where humans respond to action preference queries in the driving domain, and demonstrate our algorithm can be run on a real robot arm for a reaching task.

Baselines. We compare our approach against 2 baselines. *Random.* We present a random baseline policy that queries the top two actions in terms of Q-value with a fixed probability. In continuous action-space settings, the second-best action is not well-defined, so we instead select random actions to query as well.

Uncertainty. The uncertainty baseline uses a method similar to the novelty heuristic in ThriftyDAgger [8] to decide when to query. In particular, we present a query when the novelty of the current state is high (the variance of the Q-value of the best action integrating across $\omega \mid \Omega, \mathcal{D}$ exceeds a threshold). Unlike ThriftyDAgger [8], we cannot request full demonstrations in our setting. We instead query the top two actions by expected Q-value when our approach decides to query. Again, in continuous action-space settings, the second-best action is not well-defined, so instead we select actions by the amount they reduce the uncertainty of the Q-value of the action being selected.

To compare approaches in simulation, we simulate an expert policy which assumes access to the task representation as a proxy for a human expert which can answer queries made by the robot. We hypothesize that our approach will outperform baseline approaches in terms of the tradeoff between the task performance and the number of queries.

A. Driving Simulation Experiments

We conduct experiments in a simulated driving environment [52] where the goal is to control a car driving on a crowded highway, and the action space consists of 5 discrete actions: $A = \{\text{shift left}, \text{shift right}, \text{slower}, \text{faster}, \text{idle}\}$. The tasks (different ω) in this setting correspond to different preferences over the desired lane and lane changes, speed range, acceleration, and following distance. We train our policies across these tasks using DQNs [49], [53].

We compared our approach against baselines in this environment across random different task initializations in Figure 2, using a simulated human expert modeled by a DQN with access to the task ω. We tuned the c parameter in simulation to ensure our approach and baselines made similar numbers of queries for fairness. An example of a decision made by our method is shown in Figure 3.

We then vary the querying threshold for each method (e.g., c for ours) to analyze the tradeoff between number of queries and the performance (score) in the task. Pareto frontiers showing this tradeoff across different querying parameters are presented in Figure 6, demonstrating one can tune the c constant of our method to set this tradeoff. These Pareto frontiers suggest our approach performs robustly and outperforms baselines.

B. Driving User Studies

With IRB approval from Stanford University Research Compliance Office, we repeated the driving experiments on real humans instead of simulated experts.

Experimental Setup. We conducted our experiment using an online web interface. Subjects completed a pre-experiment survey in which they stated their gender, preferred lane,

[1] To focus more on the other experiments where we work with real human users or a real robot, we present the results of the GridWorld simulations in Appendix B, which can be found on the website.

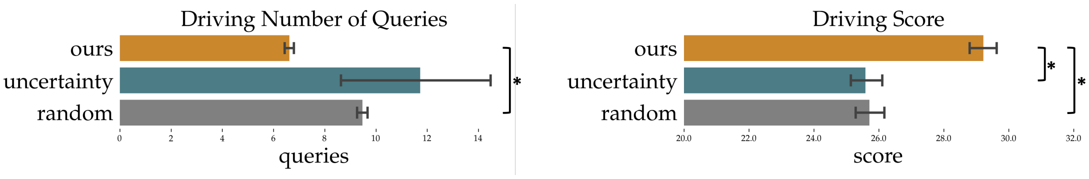

Fig. 2: In the driving environment, our approach needs fewer queries on average than either baseline, while significantly outperforming both of them in average score across tasks.

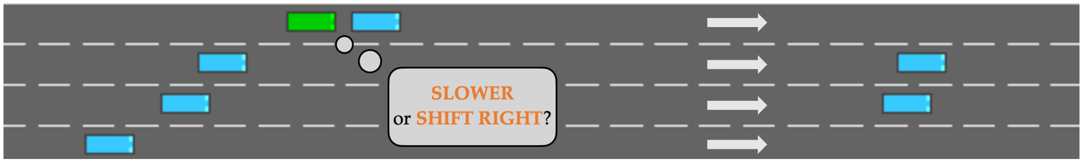

Fig. 3: The ego car (green) is in a difficult situation where its desired speed is higher than the car in the front, but it cannot shift further left. A human with aggressive preferences may prefer to shift right one lane and try to overtake the car in front, while a less aggressive human would likely prefer slowing down. Since these different preferences dramatically affect the desirability of each approach, not knowing how aggressive the human driver is, our approach queries for their preference between slowing down and overtaking by turning right, updating its belief about the human based on their response.

TABLE I: User Study Results

	Subjective Driving Metrics					Objective Driving Metrics			
Method	Important Points	Reasonable Options	Intelligent Questions	Adapted Well	Drove Well	Speed Non-Adherence	Lane Adherence	Number of Queries	Repetitive Questions
Ours	**5.32±0.26**	**4.95±0.23**	**4.86±0.18**	**5.32±0.19**	**5.36±0.20**	**7.94±0.68**	**0.56±0.10**	**8.24±0.40**	**0.00±0.00**
Uncertainty	4.00±0.42	**4.82±0.36**	**4.27±0.42**	**5.05±0.39**	**4.73±0.41**	14.7±2.2	**0.51±0.08**	10.33±0.31	8.51±0.29
Random	4.23±0.30	3.41±0.31	3.55±0.33	4.23±0.32	4.32±0.36	11.3±1.6	0.47±0.05	**8.81±0.33**	1.03±0.10

and preferred speed (between 20 m/s and 30 m/s). We additionally asked the subjects to state whether they preferred their car to adhere to their speed preferences or lane preferences more. Subjects were told to express consistent preferences, and, at the end, were asked to subjectively rate the effectiveness of each algorithm.

Procedure. We gathered data from 22 subjects (11 female, 10 male, 1 preferred not to answer), and each subject responded to queries online over the course of 5 trajectories for each algorithm (our approach or one of the two baselines, unknown to the subjects) in randomized order.

Independent Variables. The querying method used, either our approach or one of the two baselines, is the independent variable.

Dependent Measures. Subjective measures are the evaluations performed by the subjects after each algorithm: a seven-point scale survey of how much they agree with the statements: “The car asked questions at the important decision points,” “The questions included choices between reasonable options,” “Overall, the questions seemed intelligently timed and picked,” “Over time, the car adapted to my driving preferences,” and “Overall, the car drove in the way I wanted.”

Objective measures include the speed non-adherence, the percentage relative difference between the average speed of the car and the subject’s preferred speed; lane adherence, the proportion of the time the car stayed in the subject’s preferred lane; as well as the number of queries needed by the algorithm and the number of repetitive questions asked, defined as consecutive steps where the algorithm asked the same question. Since adhering to speed preferences and lane preferences often conflict (when strongly adhering to a lane, speed is strongly determined by the other cars in that lane), we only included the subjects who preferred speed or lane adherence in the respective metric computations.

Hypotheses. (1) Our approach will more efficiently and better capture user preferences than baselines, indicated by obtaining higher scores on the subjective dependent measures. (2) Our approach will perform objectively better than baselines, indicated by greater speed adherence and lane adherence, and also quantitatively be more query efficient, indicated by using a lower number of queries and making fewer repetitive queries.

Results. Table I shows the different subjective assessments of the various approaches given by users with standard errors. The best performing approaches for each metric (highest or lowest contextually) up to statistical significance are bolded (Wilcoxon signed-rank with a p-value of 0.05). Our approach achieves the best performance across all metrics, and is statistically significantly better for every metric in comparison to random, and statistically significantly better compared to uncertainty for the “Important Points” metric, supporting **Hypothesis 1**. Table I also shows the different objective assessments with standard errors. The best approaches are again bolded similarly. Our approach achieves the best performance across all metrics, supporting **Hypothesis 2**.

C. Real Robot Block Pushing Experiments

We first consider a simulated environment where a robot arm must be used to push a block to a goal location. We

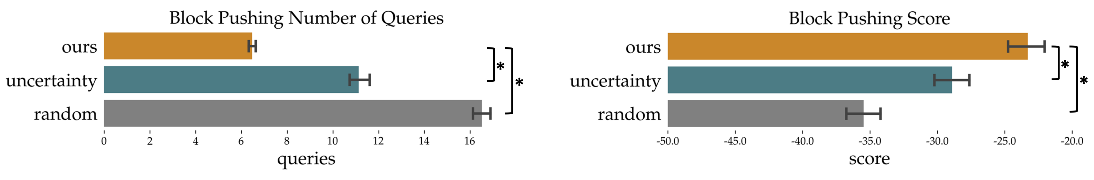

Fig. 4: Our approach makes statistically significantly fewer queries on average than all the baseline approaches in block-pushing simulation, while significantly outperforming them in average score across tasks.

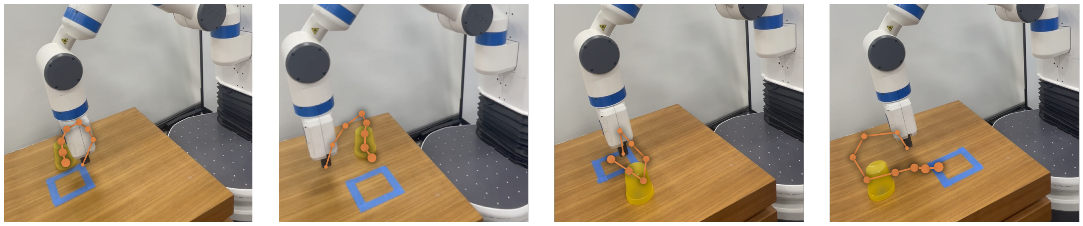

Fig. 5: Example trajectories generated by our method on the Fetch robot with a simulated human expert. The goal location (blue tape square) is observed by the simulated expert but not our approach.

view the block location as the task ω in this setting, so to succeed in this setting, the robot must query the expert to gain information about the goal location. Since the action space is continuous, we use a trained soft actor–critic policy [53], [54] as described in Section III-A. We conducted experiments in simulation using simulated expert policies, and found that our approach outperformed baselines in score while needing fewer queries (see Figure 4). We additionally constructed Pareto frontiers in Figure 6 modeling the tradeoff between queries made and scores achieved for all methods, similar to driving simulations. These again show our method robustly outperforms baselines when asking similar numbers of questions. See Appendix D for the experiment details on the block pushing setting.

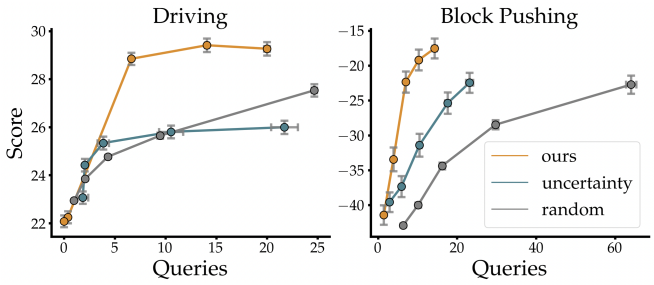

Fig. 6: Pareto frontiers comparing approaches for the driving and block pushing environment. Error bars are one standard error.

Sim2Real Transfer. We then ran the approach on a real Fetch robot arm [55], using policies trained in the simulation. On initializations where our method succeeded in simulation, we were also able to transfer to the real robot and push the yellow cup to an arbitrary goal location, as shown in Figure 5. Further details about this experiment and the videos are on the website.

V. Discussion

Summary. In this paper, we proposed an approach that designs easy-to-answer pairwise action preference queries in an online fashion. Our work maximizes the expected value derived from the queries' information and asks questions from humans at the most informative points of time. We demonstrated our approach in simulated GridWorld, a driving simulation, and a pushing task with a real robot. Across all these settings our approach outperformed baselines while needing fewer questions.

Limitations and Future Work. One limitation of this work is the potential difficulty of answering preferences between actions. While the human response model in Equation (1) accounts for uncertainty in distinguishing similarly-valued actions, there is a challenge in effectively representing queries for humans in high DoF action spaces. In discrete settings where actions are easily interpretable, this is not an issue. In continuous settings such as most robot manipulation tasks, it can be harder for humans to interpret and compare actions without good representations—a potential solution is to use a simulator to present visualizations of both actions to human experts. In future work, we plan to filter queries to ensure they are easily answerable with a visualization.

An additional limitation is the assumption that queries can be immediately answered. In settings that require fast responses such as urban driving environments this assumption may be difficult. A future approach to tackle this problem would be to train a dynamics model and use this model to predict EVOI and design queries multiple steps in advance, giving experts more time to respond.

In the future, we plan to apply this approach to more real-world robot tasks, such as Meta-World [56]. Additional exploration of complex simulation environments and baseline approaches would also help to validate our method.

Finally, theoretical analysis should be conducted on optimality, in particular with regard to (1) the effect of the approximations made to extend to continuous settings in Section III-B, (2) the selection of c and β hyperparameters, and (3) the competing effects of the pessimistic and optimistic myopia of our approach discussed in Section III-A.

References

[1] M. Li, A. Canberk, D. P. Losey, and D. Sadigh, "Learning human objectives from sequences of physical corrections," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2021, pp. 2877–2883.

[2] D. Hadfield-Menell, S. Milli, P. Abbeel, S. J. Russell, and A. Dragan, "Inverse reward design," *Advances in neural information processing systems*, vol. 30, 2017.

[3] E. Ratner, D. Hadfield-Menell, and A. D. Dragan, "Simplifying reward design through divide-and-conquer," *arXiv preprint arXiv:1806.02501*, 2018.

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

[5] 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)*, IEEE, 2019, pp. 8077–8083.

[6] K. Menda, K. Driggs-Campbell, and M. J. Kochenderfer, "Ensembledagger: A bayesian approach to safe imitation learning," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2019, pp. 5041–5048.

[7] R. Hoque, A. Balakrishna, C. Putterman, M. Luo, D. S. Brown, D. Seita, B. Thananjeyan, E. Novoseller, and K. Goldberg, "Lazydagger: Reducing context switching in interactive imitation learning," in *2021 IEEE 17th International Conference on Automation Science and Engineering (CASE)*, IEEE, 2021, pp. 502–509.

[8] R. Hoque, A. Balakrishna, E. Novoseller, A. Wilcox, D. S. Brown, and K. Goldberg, "Thriftydagger: Budget-aware novelty and risk gating for interactive imitation learning," in *5th Annual Conference on Robot Learning*, 2021.

[9] C. Wirth, R. Akrour, G. Neumann, J. Fürnkranz, *et al.*, "A survey of preference-based reinforcement learning methods," *Journal of Machine Learning Research*, vol. 18, no. 136, pp. 1–46, 2017.

[10] 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.

[11] M. Cakmak, S. S. Srinivasa, M. K. Lee, J. Forlizzi, and S. Kiesler, "Human preferences for robot-human hand-over configurations," in *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2011, pp. 1986–1993.

[12] K. Li, M. Tucker, E. Biyik, E. Novoseller, J. W. Burdick, Y. Sui, D. Sadigh, Y. Yue, and A. D. Ames, "Roial: Region of interest active learning for characterizing exoskeleton gait preference landscapes," in *International Conference on Robotics and Automation (ICRA)*, May 2021.

[13] E. Biyik, N. Huynh, M. J. Kochenderfer, and D. Sadigh, "Active preference-based gaussian process regression for reward learning," in *Proceedings of Robotics: Science and Systems (RSS)*, Jul. 2020.

[14] E. Biyik, 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)*, 2021.

[15] E. Biyik, M. Palan, N. C. Landolfi, D. P. Losey, and D. Sadigh, "Asking easy questions: A user-friendly approach to active reward learning," in *Proceedings of the 3rd Conference on Robot Learning (CoRL)*, 2019.

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

[17] E. Biyik and D. Sadigh, "Batch active preference-based learning of reward functions," in *Conference on robot learning*, PMLR, 2018, pp. 519–528.

[18] 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)*, 2020, pp. 10 952–10 959.

[19] N. Wilde, E. Biyik, D. Sadigh, and S. L. Smith, "Learning reward functions from scale feedback," in *Proceedings of the 5th Conference on Robot Learning (CoRL)*, 2021.

[20] S. M. Katz, A.-C. Le Bihan, and M. J. Kochenderfer, "Learning an urban air mobility encounter model from expert preferences," in *2019 IEEE/AIAA 38th Digital Avionics Systems Conference (DASC)*, IEEE, 2019, pp. 1–8.

[21] S. M. Katz, A. Maleki, E. Bıyık, and M. J. Kochenderfer, "Preference-based learning of reward function features," *arXiv preprint arXiv:2103.02727*, 2021.

[22] E. Biyik, K. Wang, N. Anari, and D. Sadigh, "Batch active learning using determinantal point processes," *arXiv preprint arXiv:1906.07975*, 2019.

[23] P. Viappiani and C. Boutilier, "Optimal bayesian recommendation sets and myopically optimal choice query sets.," in *NeurIPS*, 2010, pp. 2352–2360.

[24] R. Cohn, E. Durfee, and S. Singh, "Comparing action-query strategies in semi-autonomous agents," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 25, 2011, pp. 1102–1107.

[25] F. L. Da Silva, P. Hernandez-Leal, B. Kartal, and M. E. Taylor, "Uncertainty-aware action advising for deep reinforcement learning agents," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, 2020, pp. 5792–5799.

[26] J. Spencer, S. Choudhury, M. Barnes, M. Schmittle, M. Chiang, P. Ramadge, and S. Srinivasa, "Expert intervention learning," *Autonomous Robots*, vol. 46, no. 1, pp. 99–113, 2022.

[27] B. D. Argall, S. Chernova, M. Veloso, and B. Browning, "A survey of robot learning from demonstration," *Robotics and autonomous systems*, vol. 57, no. 5, pp. 469–483, 2009.

[28] B. Akgun, M. Cakmak, K. Jiang, and A. L. Thomaz, "Keyframe-based learning from demonstration," *International Journal of Social Robotics*, vol. 4, no. 4, pp. 343–355, 2012.

[29] A. D. Dragan and S. S. Srinivasa, *Formalizing assistive teleoperation*. MIT Press, July, 2012.

[30] S. Javdani, S. S. Srinivasa, and J. A. Bagnell, "Shared autonomy via hindsight optimization," *Robotics science and systems: online proceedings*, vol. 2015, 2015.

[31] R. P. Khurshid and K. J. Kuchenbecker, "Data-driven motion mappings improve transparency in teleoperation," *Presence*, vol. 24, no. 2, pp. 132–154, 2015.

[32] W. B. Knox and P. Stone, "Tamer: Training an agent manually via evaluative reinforcement," in *2008 7th IEEE international conference on development and learning*, IEEE, 2008, pp. 292–297.

[33] C. Celemin and J. Ruiz-del Solar, "Coach: Learning continuous actions from corrective advice communicated by humans," in *2015 International Conference on Advanced Robotics (ICAR)*, IEEE, 2015, pp. 581–586.

[34] R. Arakawa, S. Kobayashi, Y. Unno, Y. Tsuboi, and S.-i. Maeda, "Dqn-tamer: Human-in-the-loop reinforcement learning with intractable feedback," *arXiv preprint arXiv:1810.11748*, 2018.

[35] 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*, vol. 32, 2018.

[36] A. Bajcsy, D. P. Losey, M. K. O'Malley, and A. D. Dragan, "Learning from physical human corrections, one feature at a time," in *Proceedings of the 2018 ACM/IEEE International Conference on Human-Robot Interaction*, 2018, pp. 141–149.

[37] S. Jamieson, J. P. How, and Y. Girdhar, "Active reward learning for co-robotic vision based exploration in bandwidth limited environments," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2020, pp. 1806–1812.

[38] V. Myers, E. Biyik, N. Anari, and D. Sadigh, "Learning multimodal rewards from rankings," in *Conference on Robot Learning*, PMLR, 2022, pp. 342–352.

[39] M. Tucker, E. Novoseller, C. Kann, Y. Sui, Y. Yue, J. W. Burdick, and A. D. Ames, "Preference-based learning for exoskeleton gait optimization," in *2020 IEEE international conference on robotics and automation (ICRA)*, IEEE, 2020, pp. 2351–2357.

[40] C. Basu, E. Bıyık, Z. He, M. Singhal, and D. Sadigh, "Active learning of reward dynamics from hierarchical queries," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2019, pp. 120–127.

[41] D. J. Hejna III and D. Sadigh, "Few-shot preference learning for human-in-the-loop rl," in *Proceedings of the 6th Conference on Robot Learning (CoRL)*, 2022.

[42] S. Habibian, A. Jonnavittula, and D. P. Losey, "Here's what i've learned: Asking questions that reveal reward learning," *ACM Transactions on Human-Robot Interaction*, 2022.

[43] H. J. Jeon, S. Milli, and A. Dragan, "Reward-rational (implicit) choice: A unifying formalism for reward learning," *Advances in Neural Information Processing Systems*, vol. 33, pp. 4415–4426, 2020.

[44] M. Zimmer, P. Viappiani, and P. Weng, "Teacher-student framework: A reinforcement learning approach," in *AAMAS Workshop Autonomous Robots and Multirobot Systems*, 2014.

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

[46] D. S. Brown, W. Goo, and S. Niekum, "Better-than-demonstrator imitation learning via automatically-ranked demonstrations," in *Conference on robot learning*, PMLR, 2020, pp. 330–359.

[47] X. Chen, P. N. Bennett, K. Collins-Thompson, and E. Horvitz, "Pairwise ranking aggregation in a crowdsourced setting," in *Proceedings of the sixth ACM international conference on Web search and data mining*, 2013, pp. 193–202.

[48] L. Chen, H. Hassani, and A. Karbasi, "Near-optimal active learning of halfspaces via query synthesis in the noisy setting," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 31, 2017.

[49] 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.

[50] V. Konda and J. Tsitsiklis, "Actor-critic algorithms," *Advances in neural information processing systems*, vol. 12, 1999.

[51] I. Grondman, L. Busoniu, G. A. Lopes, and R. Babuska, "A survey of actor-critic reinforcement learning: Standard and natural policy gradients," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 42, no. 6, pp. 1291–1307, 2012.

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

[53] A. Raffin, A. Hill, A. Gleave, A. Kanervisto, M. Ernestus, and N. Dormann, "Stable-baselines3: Reliable reinforcement learning implementations," *Journal of Machine Learning Research*, vol. 22, no. 268, pp. 1–8, 2021.

[54] T. Haarnoja, A. Zhou, K. Hartikainen, G. Tucker, S. Ha, J. Tan, V. Kumar, H. Zhu, A. Gupta, P. Abbeel, *et al.*, "Soft actor-critic algorithms and applications," *arXiv preprint arXiv:1812.05905*, 2018.

[55] M. Wise, M. Ferguson, D. King, E. Diehr, and D. Dymesich, "Fetch and freight: Standard platforms for service robot applications," in *Workshop on autonomous mobile service robots*, 2016.

[56] T. Yu, D. Quillen, Z. He, R. Julian, K. Hausman, C. Finn, and S. Levine, "Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning," in *Conference on robot learning*, PMLR, 2020, pp. 1094–1100.

[57] D. Y.-T. Hui, M. Chevalier-Boisvert, D. Bahdanau, and Y. Bengio, "Babyai 1.1," *arXiv preprint arXiv:2007.12770*, 2020.

[58] M. E. Harmon and S. S. Harmon, "Reinforcement learning: A tutorial.," 1997.

[59] M. Plappert, M. Andrychowicz, A. Ray, B. McGrew, B. Baker, G. Powell, J. Schneider, J. Tobin, M. Chociej, P. Welinder, *et al.*, "Multi-goal reinforcement learning: Challenging robotics environments and request for research," *arXiv preprint arXiv:1802.09464*, 2018.

Appendix

In the appendix, we present details about our approach and experiments. See http://tinyurl.com/online-active for a website summarizing our results.

A. *Derivation*

We adopt the setting in Section III. We define the set of past query responses to be $\mathcal{D} = \Big((a_{i_1}^{(1)}, s^{(1)}, (a_1^{(1)}, a_2^{(1)})), (a_{i_2}^{(2)}, s^{(2)}, (a_1^{(2)}, a_2^{(2)})), \dots, (a_{i_n}^{(n)}, s^{(n)}, (a_1^{(n)}, a_2^{(n)})) \Big)$ for each $i_\bullet \in \{1, 2\}$, and denote $\mathcal{D}_1 = \mathcal{D} \cup \{(a_1, s, (a_1, a_2))\}$, $\mathcal{D}_2 = \mathcal{D} \cup \{(a_2, s, (a_1, a_2))\}$ for fixed actions (a_1, a_2) to potentially query in state s. The EVOI is defined as the expected change in the value function $V(s;\omega) = \max_a Q^*(s, a; \omega)$ of the current state after asking a query and updating our belief state about the policy Q function. Assuming optimality of our learned $\max_a Q^\pi(s, a; \omega)$, we obtain the following formula for the EVOI of the query (a_1, a_2).

$$\begin{aligned}
\mathrm{EVOI}(a_1, a_2) &= \mathop{\mathbb{E}}_{\text{query response}} \left[\mathop{\mathbb{E}}_{\omega|\text{after query}} V(s;\omega) \right] - \mathop{\mathbb{E}}_{\omega|\text{before query}} V(s;\omega) \\
&= \left[\mathop{\mathbb{E}}_{\omega|\Omega,\mathcal{D}} H^{\pi^*}(a_1, a_2, s, \omega) \right] \mathop{\mathbb{E}}_{\omega|\Omega,\mathcal{D}_1} V(s;\omega) + \left[\mathop{\mathbb{E}}_{\omega|\Omega,\mathcal{D}} H^{\pi^*}(a_2, a_1, s, \omega) \right] \mathop{\mathbb{E}}_{\omega|\Omega,\mathcal{D}_2} V(s;\omega) - \mathop{\mathbb{E}}_{\omega|\Omega,\mathcal{D}} V(s.;\omega) \\
&= \mathop{\mathbb{E}}_{\omega|\Omega,\mathcal{D}} \Big[H^\pi(a_1, a_2, s, \omega) \mathop{\mathbb{E}}_{\omega'|\Omega,\mathcal{D}_1} [V(s;\omega')] + H^\pi(a_2, a_1, s, \omega) \mathop{\mathbb{E}}_{\omega'|\Omega,\mathcal{D}_2} [V(s;\omega')] - V(s;\omega) \Big] \\
&= \mathop{\mathbb{E}}_{\omega|\Omega,\mathcal{D}} \Big[H^\pi(a_1, a_2, s, \omega) \mathop{\mathbb{E}}_{\omega'|\Omega,\mathcal{D}_1} \max_a Q^\pi(s, a; \omega') \\
&\qquad + H^\pi(a_2, a_1, s, \omega) \mathop{\mathbb{E}}_{\omega'|\Omega,\mathcal{D}_2} \max_a Q^\pi(s, a; \omega') - \max_a Q^\pi(s, a; \omega) \Big]
\end{aligned}$$

B. *GridWorld Experiments*

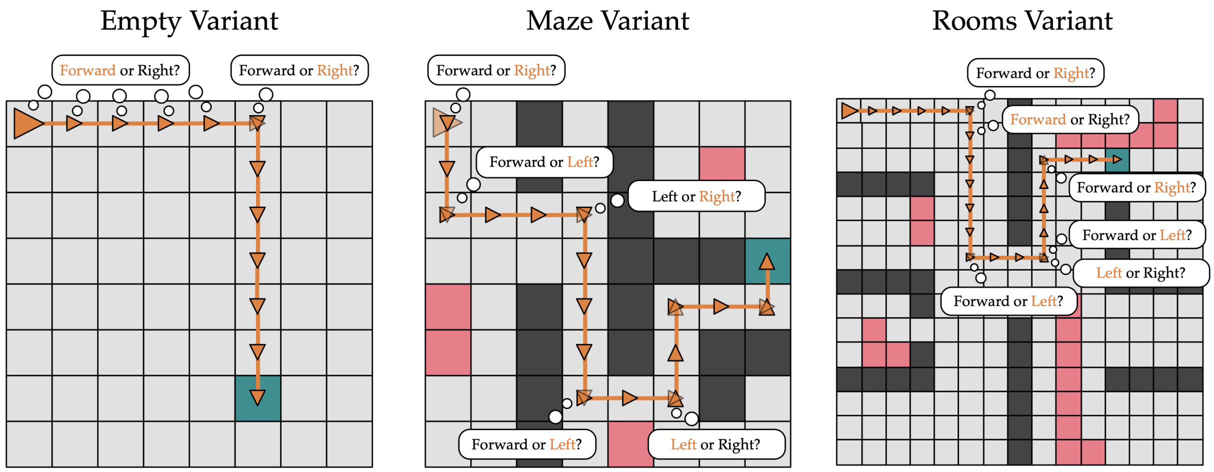

Fig. 7: GridWorld variants studied. Agent start position and orientation are shown by the large orange arrow. Empty spaces are gray, walls are black, and lava is red. The goal location is randomized uniformly across empty spaces at the start of each episode. In the diagrams above, we show a trajectory of our approach on each environment for a randomly-selected goal location, as well as the queries made by our method (with expert responses highlighted in orange).

Our initial results test our algorithm in a GridWorld environment [57]. This environment involves an agent navigating a grid to reach a goal destination. In this setting, our task distribution Ω is a uniform distribution over possible valid goal locations. We train a policy conditioned on goal locations $\omega \in \Omega$ to navigate these environments using a tabular model trained with value iteration [58]. We then compare our querying approach against the baseline approaches described above, using the learned policy conditioned on the true goal as our expert policy. We focus on the following three GridWorld environment variants, shown in Figure 7: **Empty Variant:** The agent starts in the upper left corner of a 8x8 GridWorld and navigates to a random goal location within the grid. **Maze Variant:** The agent starts in the upper left corner of a 8x8 GridWorld and navigates to a random goal location within the grid, avoiding a maze of walls and lava. **Rooms Variant:** The agent starts in the upper left corner of a 15x15 GridWorld environment, partitioned into multiple room sections with walls and lava, and navigates to a random grid location.

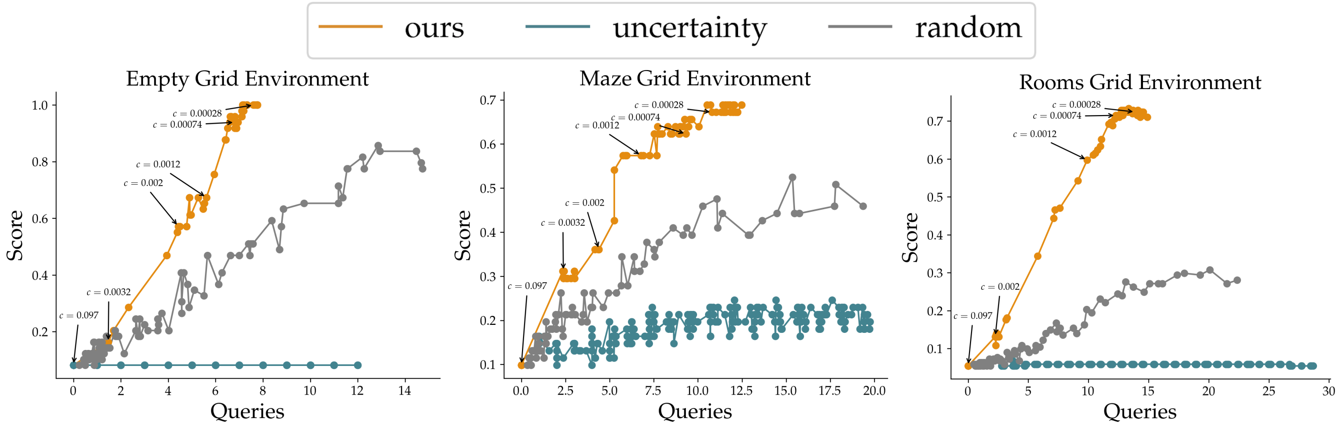

Fig. 8: Pareto-frontiers of methods on the GridWorld environment variants. Each point in the plots represents the number of queries made and average score of running a method for a particular initialization of querying parameters. By varying the querying thresholds for each approach, we see the trade offs between score and number of questions asked. For our approach, we additionally label the values of c used to generate some points to compare robustness to different c values across GridWorld settings.

We evaluate our approach against baselines in these environments by comparing the Pareto frontiers of their average performance in terms of score achieved and average number of queries made per trajectory. We vary the expected number of queries made by our approach and baselines by varying the querying parameters of each method (c for our approach). By plotting the Pareto frontiers of our approach against baselines, we can compare for a fixed number of queries made by an approach, the expected score achieved in an environment. These Pareto frontiers are presented in Figure 8.

Our approach generally outperforms both baseline approaches, achieving higher score (y-axis value) for given average numbers of questions asked (x-axis value). Interestingly, the uncertainty baseline performs significantly worse than the random baseline in this setting. This performance is likely due to to the fact that in a GridWorld trajectory, even though there is high uncertainty over the goal location, all policies are able to get roughly similar Q values since the environment is fully solvable, resulting in queries that may be strongly disconnected from imp ortant decision points. Since the uncertainty querying is deterministic, unlike random, it can then get in states where it needs to ask a question but never will, resulting in very poor performance.

We additionally label some of the points in the Pareto frontier for our approach with the c values used to generate them. By comparing the location of the points with the same value of c across GridWorld settings we can see the value of c transfers across similar settings. For instance, in all three settings, once c drops below ≈ 0.0012 the agent starts to exceed a score of 0.5 and use 5-10 queries. Thus, similar c values tend to result in similar querying behavior and performance across different GridWorld settings.

The action space in the GridWorld environment contains three actions: $A = \{\text{turn left}, \text{turn right}, \text{move forward}\}$. The agent cannot move through wall spaces, and the episode immediately ends upon moving into a lava square. To compute parteo-frontiers, we vary the querying parameters for our three approaches as follows:

Ours: parameter varies from 10^{-4} to 10^{-1} with a step size of $\log(1.05)$ in log space.
Uncertainty: parameter varies from 10^{-4} to 10^{1} with a step size of $\log(1.05)$ in log space.
Random: parameter varies from 0.05 to 0.5 with a step size of $\log(1.05)$ in log space.

Across experiments, we use a human response precision parameter of $\beta = 10$. We used value iteration to directly solve for optimal policies across tasks, using $\gamma = 0.99$ and a fixed horizon of 50 steps. These and all experiments in this paper were run using `Intel(R) Xeon(R) Silver 4114 CPU @ 2.20GHz` processors.

C. *Driving Details*

We use a human response precision parameter of $\beta = 10$ across all experiments. For the the driving environment experiments in Section IV-A, we use $c = 0.05$ for our approach, a querying probability of 0.2 for the random baseline, and an uncertainty threshold of 46 for the uncertainty baseline. These hyperparameters were tuned so that the approaches ask similar numbers of questions to better facilitate comparisons of our approach against baselines.

Our DQNs for use by agents and expert policies were trained across a random distribution of tasks, with 100 tasks with trained DQN policies used for controlling agents and the remaining 487 tasks and trained DQN policies used as evaluation tasks and human expert simulators respectively. We trained our DQNs for 60,000 steps with a batch size of 32, a replay buffer of size 15,000, an Adam learning rate of $5 \cdot 10^{-4}$, $\gamma = 0.8$, and a two-hidden-layer, 256 unit MLP architecture.

1) Pareto Frontiers: We additionally extend the driving experiments to construct full Pareto frontiers modelling the full relationship between the number of questions asked and the score achieved by each method. As in Appendix B, we vary the querying parameter of each approach to generate points on this frontier. The resulting frontier is presented in Figure 6. We see our approach outperforms baselines in terms of score across different numbers of queries made.

We varied the querying parameters of the approaches over the following values:

Ours: 0.002, 0.01, 0.05, 0.25, and 1.25.
Uncertainty: 30.67, 46, 69, 103.5, and 155.25.
Random: 0.025, 0.05, 0.1, 0.2, and 0.4.

D. Block Pushing Details

We use a human response precision parameter of $\beta = 10$, a threshold of $c = 0.03$ for our approach, a querying probability of 0.25 for the random baseline, and an uncertainty threshold of 0.01 for the uncertainty baseline across the Fetch block pushing experiments.

We trained expert policies using soft actor-critic for the block pushing task. We trained these policies in a simulated Fetch block pushing environment [59]. To apply our approaches to this setting, we used the generalization to continuous action spaces described in Section III-B.

We learned a general policy parameterized by the block's current position as well as the goal position using soft actor-critic with a 2-hidden-layer 256 unit MLP architecture. These policies required the current location of the block to be in the environments observation space, so we used the simulator to keep track to model the block's location when deploying our approaches.

Using waypoints generated by the simulator, we transferred trajectories from simulation to the real Fetch robot.

1) Pareto Frontiers: We allowed querying parameters to vary over the following values for each approach to generate this plot:

Ours: 0.0075, 0.015, 0.03, 0.06, and 0.12.
Uncertainty: 0.0025, 0.005, 0.01, 0.02, and 0.04.
Random: 0.111, 0.167, 0.25, 0.375, and 0.563.