2020 IEEE International Conference on Robotics and Automation (ICRA)
31 May - 31 August, 2020. Paris, France

Interactive Reinforcement Learning with Inaccurate Feedback

Taylor A. Kessler Faulkner[1] Elaine Schaertl Short[2] Andrea L. Thomaz[3]

***Abstract*— Interactive Reinforcement Learning (RL) enables agents to learn from two sources: rewards taken from observations of the environment, and feedback or advice from a secondary critic source, such as human teachers or sensor feedback. The addition of information from a critic during the learning process allows the agents to learn more quickly than non-interactive RL. There are many methods that allow policy feedback or advice to be combined with RL. However, critics can often give imperfect information. In this work, we introduce a framework for characterizing Interactive RL methods with imperfect teachers and propose an algorithm, Revision Estimation from Partially Incorrect Resources (REPaIR), which can estimate corrections to imperfect feedback over time. We run experiments both in simulations and demonstrate performance on a physical robot, and find that when baseline algorithms do not have prior information on the exact quality of a feedback source, using REPaIR matches or improves the expected performance of these algorithms.**

I. Introduction

In order to learn new skills, robots must be able to take input from the environment, either from sensors or from human teachers. However, input from the environment is not always correct. Sensors can intermittently fail, and teachers can be wrong. However, it is difficult for robots to predict ahead of time exactly how often, and in what ways, advice or feedback will be incorrect. If robots can use task information to model when and how feedback is incorrect, they can ignore incorrect feedback and learn from correct feedback.

We propose an algorithm, Revision Estimation from Partially Incorrect Resources (REPaIR), that estimates states and actions that are likely to receive incorrect feedback, and revises the feedback based on this estimation. This algorithm acts as a feedback filter for RL algorithms with a reward function and additional environmental feedback, so that the quality of feedback does not need to be known ahead of time. We show an illustration of the REPaIR framework in Fig. 1. REPaIR takes advantage of problems in which the robot has access to a correct reward function, which is important since it has been shown that without some kind of additional information there is no way to recover from a bad reward signal [1].

*This material is based upon work supported by the Office of Naval Research award numbers N000141612835 and N000141612785, National Science Foundation award numbers 1564080 and 1724157, and the NSF-GRFP under Grant No. DGE-1610403.

[1]Taylor A. Kessler Faulkner is with the Department of Computer Science, The University of Texas at Austin, Austin, TX 78712, USA taylor.k.f@utexas.edu

[2]Elaine Schaertl Short is with the Department of Computer Science, Tufts University, Medford, MA 02155, USA elaine.short@tufts.edu

[3]Andrea L. Thomaz is with the Department of Electrical and Computer Engineering, The University of Texas of Austin, Austin, TX 78712, USA athomaz@ece.utexas.edu

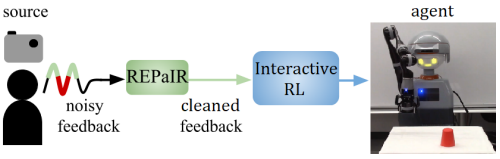

Fig. 1: REPaIR Framework

REPaIR is based on the insight that the quality of the feedback can depend on the state-action pair. For example, consider a vision system that loses sight of objects outside a certain range, or a human teacher who gets confused about what the goal of the task is. REPaIR uses the cumulative reward at the end of each learning episode to determine whether the feedback received was correct or incorrect. If feedback received reflects the cumulative reward received (positive feedback leads to higher total reward, negative feedback leads to lower total reward), the robot has a higher trust in the feedback. Otherwise, the robot has lower trust in the feedback. REPaIR then either keeps, discards, or changes the feedback based on the trust. REPaIR can be used to estimate feedback correctness for feedback-utilizing Interactive RL algorithms.

We test REPaIR with three baseline algorithms: Policy Shaping (PS) [2], and two versions of TAMER+RL (TAMER-P, TAMER-W) [3], which all have trust parameters to handle varying feedback quality. We show that adding REPaIR to these baselines matched or exceeded performance for 83.33% (TAMER-P), 83.33% (TAMER-W), and 100% (PS) of tested feedback quality settings in simulation. The average performance with REPaIR in these settings exceeded or matched more than half of mismatched trust parameters, which implies that REPaIR matches or improves expected performance on these baselines when they do not have prior information on feedback quality. We also demonstrate REPaIR on a manipulation task with a physical robot in which the robot must grasp a cup, with feedback provided by a noisy object detector. We found the differences in means consistent with the results in simulations.

II. Background

Prior work has proposed algorithms that provide some form of compensation for incorrect supplemental environmental information. These include methods using feedback, advice, and demonstrations from which robots can learn.

Like Interactive RL, Inverse Reinforcement Learning (IRL) allows agents to learn from human teachers. However, in IRL, the agent is provided with full (if incorrect) demonstrations from the teacher and has to estimate the reward function that results in these demonstrations [4]–[7]. Like our work, Evans et al. [4] observes that there are patterns to the behavior

of incorrect agents. Their work estimates the preferences of people using Bayesian inference over models of possible inaccurate agents, and assumes that there are patterns to the behavior of incorrect agents. However, all of these works require full demonstrations from users, which may be more difficult than giving feedback depending on the complexity of the task. Other methods require other ground truth information, such as a ranking over all demonstrations [5], the knowledge that all demonstrations are incorrect [6], or that bad demonstrations can be treated as sparse noise [7]. Our method does not rely on these assumptions, and allows feedback to be incorrect in any amount while using a reward function defined by an expert.

Other work combines potentially incorrect feedback ([2], [3], [8]) or demonstrations [9]–[11] with an RL framework. Griffith et al. [2] use static trust in a teacher's feedback, which may not account for varying levels of feedback performance. Knox and Stone [3] propose several methods of combining feedback with a reward function: the two top-performing methods use a weighted method of taking actions based on the human feedback, which discounts good feedback as well as bad. Our algorithm can learn how good feedback is over time, rather than setting a trust parameter before receiving feedback. Sridharan [8] keeps track of multiple policies from a reward function, and one policy from feedback. The trust in the feedback is weighted based on the amount of agreement of actions between its policy and the reward function policies. However, this method stores $N+1$ policies, as the reward function is not assumed to be correct. This can take a large amount of memory as the state space size grows. Our method, REPaIR, uses less space since fewer policies are stored. One method similar to [8] also bases the trust in a teacher based on a comparison between the advice from a teacher and the currently learned Q-values, as well as the current trust in a deep RL algorithm [12]. Our work uses feedback (information about the quality of an action *after* it is taken) rather than advice (information about the quality of potential actions *before* they are taken), as the robot would take incorrect advice when seen for the first time, but feedback can be considered first, compared to the received reward, and worked into the estimation of potentially incorrect information.

Other prior work attempts to detect unreliable sensors [13], [14]. However, these methods require multiple sensors, which may not be available on publicly deployed robots. Our algorithm can take only one unreliable feedback source and a sparse reward function to estimate what feedback is incorrect.

III. Algorithm

A. Framework for Interactive RL with Inaccurate Feedback

Along with the REPaIR algorithm, we also propose a framework for discussing Interactive RL problems with inaccurate feedback. As with the standard RL problem, we begin with a Markov Decision Process (MDP). This framework is inspired by Everitt et al. [1], who developed a framework for CRMDPs (Corrupt Reward MDPs), for which the reward signal itself is unreliable, and thus a *corruption function* between correct and received reward is defined. We propose the following Imperfect Feedback MDP (IFMDP), consisting of a tuple $(S, A, T, R, F^*, F, \gamma)$ as follows:

- S : a set of states
- A : a set of actions
- $T(s, a, s')$: probabilities of transitioning to $s' \in S$ when taking $a \in A$ in $s \in S$
- $R(s, a)$: the reward function
- $F^*(s, a), F(s, a)$: a "correct" and given feedback function, related by Γ s.t. $F(s, a) = \Gamma(F^*(s, a))$
- γ: a discount factor

S, A, T, R and γ are all identical to the standard MDP. We add F^* and F, which are the correct and given feedback functions. *Correct* feedback will elicit the optimal policy, π^*, for the given reward function R. That is, more favorable actions in π^* will receive higher feedback. The *corruption function*, Γ, is the difference between $F^*(s, a)$ and $F(s, a)$ s.t. $F(s, a) = \Gamma(F^*(s, a))$, as defined by Everitt et al. to relate reward signals [1].

B. Preliminaries

In this work, we take advantage of a correct reward function R. Otherwise, with an imperfect teacher and without additional information, it is impossible for the agent to tell what behavior is actually desired, as shown in [1]. We use sparse reward functions, as consistent feedback is most helpful when the reward function is not meticulously shaped to guide the agent to the goal. Our R has positive rewards only on goal states and low negative rewards only on states to be avoided, potentially with small negative rewards on all other state-action pairs if fast travel to goal states should be encouraged. Such reward functions are easy for people to define, compared to dense reward functions. However, they are also more difficult for agents to learn from than well-populated rewards, as high-magnitude rewards will take longer to propagate through the large areas of small negative reward. Thus F^* acts as a dense representation of R, as the agent can receive meaningful ranked feedback on each state-action pair.

We leverage a correct reward function R to compensate for incorrect feedback. To do so, we observe that the current state and action has some impact on the correctness of the feedback. Some prior work has assumed that feedback either improves or worsens over time, or that feedback is randomly incorrect some percentage of the time [2], [3]. In practice, the current state-action pair often has an effect on the correctness of feedback. For example, consider a human teacher that does not understand the joint limits of a robotic arm. In this case, states and actions near the limits can receive incorrect feedback, as the teacher may give feedback that suggests an efficient path that takes the robotic arm into unsafe or impossible joint positions. However, other states and actions receive correct feedback. Another example is a vision system that can reliably detect distinct objects, but not if two such objects are too close to each other. In this case, states in which two or more objects are close may receive incorrect feedback on actions intended to grasp one object.

C. REPaIR Algorithm

We present Revision Estimation from Partially Incorrect Resources (REPaIR) in Algorithm 1. REPaIR gives an estimation of the inverse of the corruption function, Γ, to an Interactive RL algorithm to compensate for bad feedback. We will refer to this estimation as Γ', where the input to Γ' is $F(s,a)$, a state-action pair (s,a) with feedback f. The output of Γ' is an estimated revision f', an attempt to recreate F^*. Γ cannot be directly measured, as there is no ground truth for correct feedback until the value function is learned, at which point revisions to feedback are unnecessary. Γ' gives corrected feedback to a learning algorithm.

To update Γ' (and thus its estimate of the inverse of Γ), the agent uses the only ground-truth feedback, the reward function R. Since immediate high rewards do not always indicate the highest cumulative reward when the goal is reached, especially in sparse reward functions, cumulative rewards collected at the end of each episode are used as ground-truth information.

As the agent learns, it saves the state-action trajectory ξ that it takes in each learning episode, with feedback/advice f_i for each (s_i, a_i). At the end of each episode, it saves its final total reward, R_ξ. For $(s_i, a_i, f_i) \in \xi$, if a higher R_ξ has not been seen for (s_i, a_i, f_i), we save it in the highest rewards: $R_{max}[(s_i, a_i, f_i)] = R_\xi$. We assign a trust t_i in the range $[0, 1]$ to the feedback on (s_i, a_i, f_i):

$$t_{(s_i,a_i,f_i)} = \begin{cases} \frac{R_{max}[(s_i,a_i)]-min(R_{max})}{\max(R_{max})-\min(R_{max})} & f_i \text{is positive} \\ 1 - \frac{R_{max}[(s_i,a_i)]-min(R_{max})}{\max(R_{max})-\min(R_{max})} & f_i \text{is negative} \end{cases} \quad (1)$$

The intuition behind this trust assignment is that an action should not be catastrophic if it results in one of the higher seen cumulative rewards. REPaIR determines whether to invert, keep, or discard feedback as follows, where t_{min} and t_{max} are threshold parameters. If $t_{(s_i,a_i,f_i)} \geq t_{max}$, REPaIR keeps the feedback: $f_i = f_i$. If $t_{(s_i,a_i,f_i)} \leq t_{min}$, REPaIR inverts the feedback: $f_i = -f_i$. Otherwise, REPaIR discards the feedback: $f_i = 0$.

Algorithm 1: REPaIR

```
Γ' = feedback revision estimator;
R_max = −∞ = maximum total rewards seen;
t_min, t_max = thresholds for inverting and keeping feedback;
while learning do
    R_ξ = 0;
    ξ = [];
    while episode not over do
        s_i, a_i = current state, action;
        R_ξ = R_ξ + reward;
        f_i = feedback;
        ξ.append([s_i, a_i, f_i])
    end
    for (s_i, a_i, f_i) ∈ ξ do
        R_max[(s_i, a_i, f_i)] = max(R_max[(s_i, a_i, f_i)], R_ξ);
        t_(s_i,a_i,f_i) = { (R_max[(s_i,a_i)] − min(R_max)) / (max(R_max) − min(R_max))   f_i is positive
                          { 1 − (R_max[(s_i,a_i)] − min(R_max)) / (max(R_max) − min(R_max))   f_i is negative ;
        if t_(s_i,a_i,f_i) ≤ t_min then
            Γ'(f_i) = −f_i;
        else if t_(s_i,a_i,f_i) ≥ t_max then
            Γ'(f_i) = f_i;
        else
            Γ'(f_i) = 0;
        end
    end
end
```

IV. Experimental Baselines

We compare against three baselines and Q-Learning [15] (RL without feedback): two versions of TAMER+RL [3], and Policy Shaping (PS) [2]. REPaIR is used to supplement the two TAMER+RL methods and PS. We chose these baseline algorithms because they take additional environmental feedback but do not use it to modify their reward or value functions, which can cause changes to the final optimal policy [16]. Thus the reward function will remain correct, which is a requirement for REPaIR. We expect REPaIR to improve the performance of these baselines because REPaIR estimates feedback quality based on the current state and action, rather than assuming a static trust [2] or simply decreasing trust over time [3]. Thus, the trust parameters for these baselines do not need to be varied to match the feedback quality, which may not be known in advance in a real-world scenario. We implemented all algorithms in Python 2.7.

A. Q-Learning

Q-Learning, an off-policy method of RL, uses a learning rate α and a discount factor γ to learn Q-values from rewards using a Bellman update over episodes. For our baseline, we use Q-learning with Boltzmann exploration [15], [17]. Using Boltzmann exploration, the probability of taking any action a in state s is

$$\Pr_q(s,a) = \frac{e^{\frac{Q(s,a)}{\tau}}}{\sum_{a'} e^{\frac{Q(s,a')}{\tau}}}$$

using the learned Q-values $Q(s,a)$ and τ, a temperature parameter. We implement all of the following baselines with Q-Learning as the underlying RL algorithm.

B. TAMER+RL

TAMER is an algorithm for replacing a reward function with human feedback [18], [19]. An extension to TAMER experimented with several methods of combining scalar feedback with a reward function [3]. We consider two such methods that were shown to outperform SARSA [20], an on-policy method of RL. In the following equations, $\hat{H}(s,a)$ is the human's reward function, learned over time using TAMER. We do make one small change to the usual TAMER feedback.

Usually, TAMER uses a learned predictor of feedback after each state-action pair, in tasks that do not allow immediate feedback from humans after each action [18], [19]. Our tasks allow immediate feedback. Thus, when feedback is given for a state-action pair, $\hat{H}(s,a)$ is this feedback. When no feedback has been given, we use the learned predictor. Using the actual current feedback rather than the predicted in all cases allows TAMER to learn from good feedback more quickly, which we can take advantage of in this environment.

1) TAMER-P: $P(a = argmax_a[\hat{H}(s,a)]) = p$, otherwise original RL agent's action selection mechanism is used. p is gradually diminished over time, so the human's feedback is more influential at the start of the learning process.
2) TAMER-W: $a = argmax_a[Q(s,a) + w{*}\hat{H}(s,a)]$

We anneal p and w over time, as in [3]. They are decreased by 0.01% at the end of each learning episode.

C. Policy Shaping

Policy Shaping [2], [21] enables people to give binary feedback, positive or negative, to a learning robot. This feedback shapes the robot's policy, influencing it towards state-action pairs that have received positive feedback from a human teacher. All actions consider the human feedback as the difference in positive and negative values given by the teacher, or $\Delta_{s,a}$. Using $\Delta_{s,a}$ rather than the count of positive feedback on (s,a) compensates for the possibility that teachers may be slightly inconsistent in their feedback on the same state-action pair at different times.

Policy Shaping affects the exploration style of the robot, rather than influencing the rewards or Q-values of the MDP directly. Let the probability of taking any action using exploration methods based purely on the MDP be $\Pr_q(a|s)$. The probability that an action is good using feedback is

$$\Pr_c(a|s) = \frac{C^{\Delta_{s,a}}}{C^{\Delta_{s,a}} + (1-C)^{\Delta_{s,a}}},$$

where $C \in [0,1]$ is a trust parameter, with 0 being complete distrust in the human teacher and 1 being complete trust. When $C = 0.5$, PS reduces to RL with no feedback. In our experiments, we cap $\Delta_{s,a}$ between -50 and 50 to avoid overflow computation errors in Python 2.7. Using $\Pr_q(a|s)$ and $\Pr_c(a|s)$, the probability of taking any action $a \in A$ in state $s \in S$ while learning is

$$\Pr_p(a|s) = \frac{\Pr_q(a|s)\Pr_c(a|s)}{\sum_{\alpha \in A}\Pr_q(\alpha|s)\Pr_c(\alpha|s)}.$$

V. Validation

We run experiments in simulation to compare the performance of interactive RL algorithms with and without REPaIR. As a proof of concept, we also run experiments on a physical robot with a Kinova Jaco 7-dof arm and Robotiq gripper to compare the performance of Policy Shaping (PS) and PS+REPaIR.

A. Simulation Task

We use a staging task in simulation, for which it is easy to modify the feedback performance to test. The agent learns to place six objects into two bins, with four objects in bin one (b_1) and two objects in bin two (b_2). The agent has sixteen objects total, and can place or remove one object at a time from the bins. The agent can also choose to end the task at any time, and must do so to end the learning episode. The MDP is as follows:

- S : [b_1 contents, b_2 contents]
- A : ["place one object into b_1", "place one object into b_2", "remove one object from b_1", "remove one object from b_2", "end task"]
- $T(s,a,s')$: deterministic transition function
- $R(s,a)$: +100 if $a =$ "end task" at goal state, -10 if $a =$ "end task" not at goal state, -1 otherwise.

We define perfect feedback (F^*) for this task as follows. The critic gives positive rewards for ending the task when there are four objects in b_1 and two objects in b_2, for adding objects to b_1 or b_2 when there are less than 4 and 2 respectively, and for removing objects from b_1 or b_2 when there are more than 4 and 2 respectively. The critic gives negative feedback otherwise.

For this task, we averaged over 50 trials of Q-Learning to optimize the parameters for area under the learning curve (the cumulative reward over all 100 episodes): $\tau = 0.5$, $\alpha = 0.8$, $\gamma = 0.8$. If the robot does not end the task before 100 actions, the episode ends with -10 reward.

B. Simulation Experiments

We run experiments adding REPaIR to three different algorithms in simulation: Policy Shaping (PS) [2], [21] and the top two performing algorithms from TAMER + RL [3] (TAMER-P, TAMER-W). We vary the feedback quality as follows. We pick some percent of states to receive correct feedback, and choose these states at random. We vary the correct percentage from 0 to 100, in increments of 20. We compare the algorithm (PS, TAMER-P, TAMER-W) to Q-Learning and the algorithm with all feedback first given to REPaIR. All feedback is binary good/bad (1,-1) to maintain consistency across all algorithms. Thus in the TAMER feedback predictor (for state-action pairs where no feedback has been received), all positive feedback predictions are mapped to +1 and negative predictions to -1. All experiments are run over 100 learning episodes and averaged over 30 trials of each type of teacher. We show the average area under the learning curve (the total reward over all 100 episodes) for each level of feedback correctness. This area is calculated using the composite trapezoidal rule.

One advantage of REPaIR is that it allows a single trust parameter to be used across a wide variety of feedback reliability, rather than requiring tuning to a specific source of feedback. Therefore, we keep all trust parameters (C,w,p) set to 0.8 for the experiments. We chose 0.8, as this value weights feedback positively but does not trust it fully. We vary the trust parameters for the baseline algorithms. This tests

whether the addition of REPaIR outperforms the baseline algorithms if the quality of the teacher is not known ahead of time. Trust parameters are varied from 0.0 to 1.0 in increments of 0.1 for all baselines, with minor exceptions. For TAMER, $w = 0.0$ and $p = 0.0$ are equivalent to Q-Learning, so these are not tested. For PS, $C = 0.5$ is equivalent to Q-Learning, and C cannot be exactly 0 or 1 as these lead to dividing by zero. Thus C is set between 0.01 and 0.99, excluding 0.5. We measure the percentage of trust settings (out of 10 total) that perform differently than the baseline plus REPaIR.

We set t_{min} and t_{max} for each baseline algorithm. Recall that feedback is inverted when the trust is less than or equal to t_{min}, and kept when the trust is great than or equal to t_{max}. While these parameters are task- and algorithm-specific, they are not specific to the amount of incorrect feedback. For TAMER-P, TAMER-W, and PS, t_{min} and t_{max} are [0.05,0.85], [0.0,0.95], and [0.05,0.35] respectively.

C. Robot Task

The agent learns to grasp a cup on a table by moving its gripper above the table in cardinal directions on a 4 by 4 grid, and reaching down to grasp when it is over the cup. The MDP is as follows:

- $S : x, y$ location of gripper in 4 by 4 grid
- A : all four cardinal directions, and attempt a grasp
- $R(s, a)$: +100 if robot successfully attempts grasp, -10 if robot unsuccessfully attempts grasp or after 16 actions, -1 otherwise.

The robot receives a reliable reward function from detecting whether its gripper is fully closed after attempting a grasp. If the gripper is not fully closed (blocked by the cup), the cup has been successfully grasped. The episode ends if the robot attempts a grasp or after a maximum of 16 actions. All experiments are 40 episodes long and averaged over 5 runs of each algorithm.

Feedback is given using the ORP object recognition and pose estimation system [22] to locate the position of the cup relative to the robot's gripper. If the gripper moves closer to the cup, the robot receives positive feedback, and receives negative feedback otherwise. When the arm gets in between the cup and the camera, the visual system may not perceive the cup and thus give negative feedback, as the gripper is not getting closer to any perceived cup position. Other situations (such as the gripper intersecting the cup view) may give incorrect positions for the cup's location. When initially tested, the gripper fully blocked the cup in 25% of states, and partially blocked the cup in another 12.5%. Additional noise came from changes over time, such as changing indoor lights, movement, and arm positions.

D. Robot Experiment

We optimized Q-Learning in simulation to optimize the parameters for area under the learning curve: $\tau = 0.1$, $\alpha = 1.0$, $\gamma = 0.9$. We test $C = 0.2$ ($PS{-}0.2$) and $C = 0.8$ ($PS{-}0.8$) as the midpoint performance of trusting or discounting all feedback. We set t_{min} and t_{max} to [0.0,0.95].

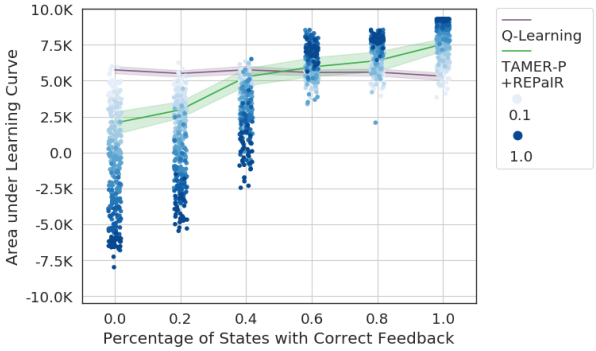

Fig. 2: TAMER-P compared to TAMER-P+REPaIR and Q-Learning. p varies for TAMER-P as shown by the varied dots, and $p = 0.8$ for TAMER-P+REPaIR

	Percentage of States with Correct Feedback					
	0.0*	0.2*	0.4*	0.6*	0.8*	1.0
Significantly Higher	**70%**	**70%**	**80%**	0%	0%	30%
Similar	10%	10%	20%	**60%**	**60%**	20%
Significantly Lower	20%	20%	0%	40%	40%	**50%**

TABLE I: Changes in performance from adding REPaIR to TAMER-P out of 10 different p settings

VI. Results

In all results discussed, +REPaIR indicates that an algorithm was run with the feedback run through Γ'. All significance values are calculated using a one-way ANOVA and a Tukey post-hoc test, with $p < 0.05$ required for significance. In Figs. 2, 3, and 4, the two lines show performance for baseline+REPaIR and Q-Learning. The baseline algorithm is represented as a gradient of points, each of which represents one run (100 episodes long), where the color represents the trust parameter setting for that run (darker is higher).

1) TAMER-P Simulation: Results for TAMER-P and TAMER-P+REPaIR are shown in Fig. 2. The percentage of p settings for which TAMER-P+REPaIR significantly outperforms or underperforms the average TAMER-P is shown in Table I. Across all feedback quality levels, adding REPaIR matched or exceeded baseline performance over the majority of p settings in 83.33% of cases (starred in Table I). TAMER-P+REPaIR also outperforms Q-Learning at 80% and 100% correct state feedback.

2) TAMER-W Simulation: Results for TAMER-W and TAMER-W+REPaIR are shown in Fig. 3. The percentage of w settings for which TAMER-W+REPaIR significantly outperforms or underperforms the average TAMER-W is shown in Table II. Over all feedback quality levels, adding REPaIR matched or exceeded baseline performance over the majority of w settings in 83.33% of cases (starred in Table II). TAMER-W+REPaIR also outperforms Q-Learning at 60%, 80% and 100% correct state feedback.

3) Policy Shaping Simulation: Results for PS and PS+REPaIR are shown in Fig. 4. The percentage of C settings for which PS+REPaIR significantly outperforms or underperforms the average PS is shown in Table III. Over

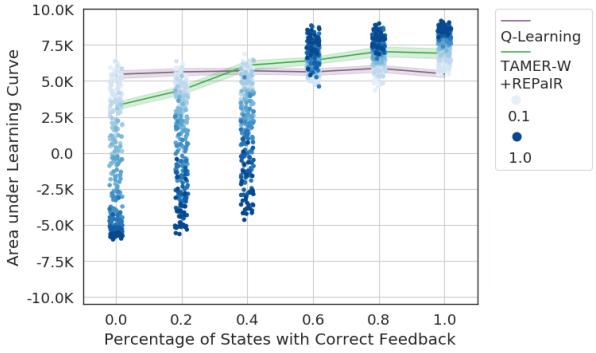

Fig. 3: TAMER-W compared to TAMER-W+REPaIR and Q-Learning. w varies for TAMER-W as shown by the varied dots, and $w = 0.8$ for TAMER-W+REPaIR

	Percentage of States with Correct Feedback					
	0.0*	0.2*	0.4*	0.6*	0.8*	1.0
Significantly Higher	**70%**	**70%**	**100%**	30%	30%	10%
Similar	10%	20%	0%	30%	**40%**	30%
Significantly Lower	20%	10%	0%	**40%**	30%	**60%**

TABLE II: Changes in performance from adding REPaIR to TAMER-W out of 10 different w settings

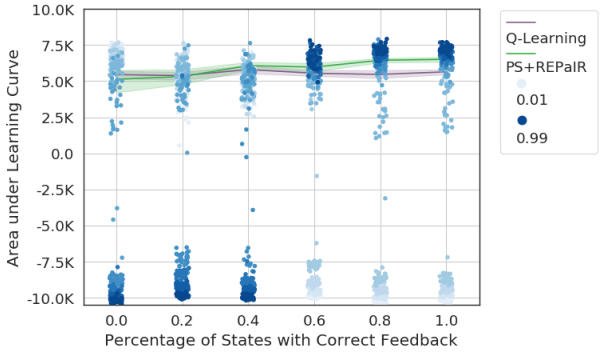

Fig. 4: PS compared to PS+REPaIR and Q-Learning. C varies for PS as shown by the varied dots, and $C = 0.8$ for PS+REPaIR

	Percentage of States with Correct Feedback					
	0.0*	0.2*	0.4*	0.6*	0.8*	1.0*
Significantly Higher	**50%**	**60%**	**100%**	**50%**	**50%**	**60%**
Similar	10%	40%	0%	20%	30%	10%
Significantly Lower	40%	0%	0%	30%	20%	30%

TABLE III: Changes in performance from adding REPaIR to PS out of 10 different C settings

all feedback quality levels, adding REPaIR adding REPaIR matched or exceeded baseline performance over the majority of C settings in 100.00% of cases (starred in Table III). PS+REPaIR also outperforms Q-Learning at 60%, 80% and 100% correct state feedback.

4) Policy Shaping on Robot: Results are shown in Fig. 5. The addition of REPaIR, with an average of 1788.3 area under the learning curve, outperforms PS with $C = 0.8$, with an average of 1680.2 area under the learning curve, and PS with $C = 0.2$, with an average of 1026.7 area under the learning curve. These results are not significant ($p = 0.44$) using Welch's Anova, but show that a physical robot can learn a task using feedback filtered through REPaIR, and suggests that using REPaIR may improve performance.

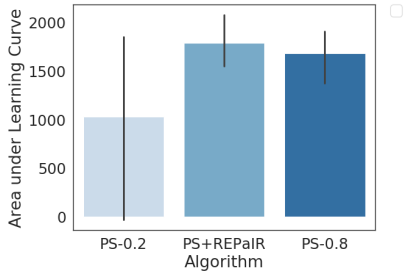

Fig. 5: Performance of PS and PS+REPaIR on a robot

VII. Discussion and Conclusions

Our simulation results show that adding REPaIR to these baselines matched or improved performance for 83.33% (TAMER-P), 83.33% (TAMER-W), and 100% (PS) of tested feedback quality settings. The average performance with REPaIR in these settings exceeded or matched more than half of mismatched trust parameters. For TAMER-P and TAMER-W, REPaIR can decrease performance when feedback is perfect, but this is balanced by the substantial performance gains for imperfect feedback. This is because over-trusting feedback lets the robot take full advantage of correct feedback, but can lead to bad performance when feedback is bad. As shown by the color gradient of baseline points in Figs. 2, 3, and 4, the best performance comes from matching trust to actual feedback quality. In contrast, REPaIR can perform well with one trust setting, thus improving learning when the quality of feedback is not fully known in advance. We demonstrated the performance of REPaIR in a real-world robot experiment, which suggested that a robot can use REPaIR to filter feedback from a sensor while learning.

Our results show that when Interactive RL algorithms do not have prior knowledge on the correctness of a feedback source, using REPaIR to estimate better quality feedback improves performance. In practice, a robot will rarely know the quality of a feedback source in advance. A human teacher might have a wide range of understanding of a task, or they may act adversarially. Sensor feedback might be highly useful for guiding learning, but might also be inconsistent in unpredictable ways. While REPaIR does have parameters that affect its performance, t_{min} and t_{max}, these parameters can be set for a certain task and baseline RL algorithm, and do not rely on the correctness of the feedback. Thus REPaIR can be used to improve Interactive RL performance when the quality of a feedback source is unknown.

Future directions for work in this area include using multimodal feedback to improve feedback evaluation. Robots could also learn to transfer learned patterns of incorrect states to new tasks, or to unseen state-action pairs by learning similarities between states. While REPaIR uses trajectories of state-action pairs, future work could extend the approach to continuous state spaces.

REFERENCES

[1] T. Everitt, V. Krakovna, L. Orseau, and S. Legg, "Reinforcement learning with a corrupted reward channel," in *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence*, 2017.

[2] S. Griffith, K. Subramanian, J. Scholz, C. L. Isbell, and A. L. Thomaz, "Policy shaping: Integrating human feedback with reinforcement learning," in *Advances in neural information processing systems*, 2013, pp. 2625–2633.

[3] W. B. Knox and P. Stone, "Combining manual feedback with subsequent mdp reward signals for reinforcement learning," in *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems: volume 1-Volume 1*. International Foundation for Autonomous Agents and Multiagent Systems, 2010, pp. 5–12.

[4] O. Evans, A. Stuhlmüller, and N. Goodman, "Learning the preferences of ignorant, inconsistent agents," in *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

[5] D. Brown, W. Goo, P. Nagarajan, and S. Niekum, "Extrapolating beyond suboptimal demonstrations via inverse reinforcement learning from observations," in *Proceedings of the 36th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, K. Chaudhuri and R. Salakhutdinov, Eds., vol. 97. Long Beach, California, USA: PMLR, 09–15 Jun 2019, pp. 783–792. [Online]. Available: http://proceedings.mlr.press/v97/brown19a.html

[6] D. H. Grollman and A. G. Billard, "Robot learning from failed demonstrations," *International Journal of Social Robotics*, vol. 4, no. 4, pp. 331–342, 2012.

[7] J. Zheng, S. Liu, and L. M. Ni, "Robust bayesian inverse reinforcement learning with sparse behavior noise," in *Twenty-Eighth AAAI Conference on Artificial Intelligence*, 2014.

[8] M. Sridharan, "Augmented reinforcement learning for interaction with non-expert humans in agent domains," in *2011 10th International Conference on Machine Learning and Applications and Workshops*, vol. 1. IEEE, 2011, pp. 424–429.

[9] M. Večerík, T. Hester, J. Scholz, F. Wang, O. Pietquin, B. Piot, N. Heess, T. Rothörl, T. Lampe, and M. Riedmiller, "Leveraging demonstrations for deep reinforcement learning on robotics problems with sparse rewards," *arXiv preprint arXiv:1707.08817*, 2017.

[20] R. S. Sutton, A. G. Barto, *et al.*, *Introduction to reinforcement learning*. MIT press Cambridge, 1998, vol. 2, no. 4.

[10] Y. Gao, J. Lin, F. Yu, S. Levine, T. Darrell, *et al.*, "Reinforcement learning from imperfect demonstrations," *arXiv preprint arXiv:1802.05313*, 2018.

[11] K. Subramanian, C. L. Isbell Jr, and A. L. Thomaz, "Exploration from demonstration for interactive reinforcement learning," in *Proceedings of the 2016 International Conference on Autonomous Agents & Multiagent Systems*. International Foundation for Autonomous Agents and Multiagent Systems, 2016, pp. 447–456.

[12] Z. Lin, B. Harrison, A. Keech, and M. O. Riedl, "Explore, exploit or listen: Combining human feedback and policy model to speed up deep reinforcement learning in 3d worlds," *arXiv preprint arXiv:1709.03969*, 2017.

[13] A. Yazidi, B. J. Oommen, and M. Goodwin, "On distinguishing between reliable and unreliable sensors without a knowledge of the ground truth," in *2015 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT)*, vol. 2. IEEE, 2015, pp. 104–111.

[14] K. Ni and G. Pottie, "Bayesian selection of non-faulty sensors," in *2007 IEEE International Symposium on Information Theory*. IEEE, 2007, pp. 616–620.

[15] C. J. Watkins and P. Dayan, "Q-learning," *Machine learning*, vol. 8, no. 3-4, pp. 279–292, 1992.

[16] A. Y. Ng, D. Harada, and S. Russell, "Policy invariance under reward transformations: Theory and application to reward shaping," in *ICML*, vol. 99, 1999, pp. 278–287.

[17] C. Watkins, "Models of delayed reinforcement learning," *PhD thesis, Psychology Department, Cambridge University*, 1989.

[18] 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.

[19] ——, "Interactively shaping agents via human reinforcement: The tamer framework," in *Proceedings of the fifth international conference on Knowledge capture*. ACM, 2009, pp. 9–16.

[21] T. Cederborg, I. Grover, C. L. Isbell, and A. L. Thomaz, "Policy shaping with human teachers," in *Twenty-Fourth International Joint Conference on Artificial Intelligence*, 2015.

[22] A. D. Allevato *et al.*, "An object recognition and pose estimation library for intelligent industrial automation," Master's thesis, University of Texas at Austin, 2016.