SAIL: Simulation-Informed Active In-the-Wild Learning

Elaine Schaertl Short
Dept. of Electrical and Computer Eng.
University of Texas at Austin
Austin, Texas, USA
elaine.short@utexas.edu

Adam Allevato
Dept. of Mechanical Eng.
University of Texas at Austin
Austin, Texas, USA
allevato@utexas.edu

Andrea L. Thomaz
Dept. of Electrical and Computer Eng.
University of Texas at Austin
Austin, Texas, USA
athomaz@ece.utexas.edu

***Abstract*—Robots in real-world environments may need to adapt context-specific behaviors learned in one environment to new environments with new constraints. In many cases, copresent humans can provide the robot with information, but it may not be safe for them to provide hands-on demonstrations and there may not be a dedicated supervisor to provide constant feedback. In this work we present the SAIL (Simulation-Informed Active In-the-Wild Learning) algorithm for learning new approaches to manipulation skills starting from a single demonstration. In this three-step algorithm, the robot simulates task execution to choose new potential approaches; collects unsupervised data on task execution in the target environment; and finally, chooses informative actions to show to co-present humans and obtain labels. Our approach enables a robot to learn new ways of executing two different tasks by using success/failure labels obtained from naïve users in a public space, performing 496 manipulation actions and collecting 163 labels from users in the wild over six 45-minute to 1-hour deployments. We show that classifiers based low-level sensor data can be used to accurately distinguish between successful and unsuccessful motions in a multi-step task ($p < 0.005$), even when trained in the wild. We also show that using the sensor data to choose which actions to sample is more effective than choosing the least-sampled action.**

***Index Terms*—in-the-wild human-robot interaction; robot learning; learning from demonstration**

I. Introduction

Co-present humans are a valuable resource for robots deployed in the wild: they can teach the robot new skills, provide feedback on existing skills, and help to ground models. Research often imagines that a human teacher is consistently, if not constantly, available to the robot for teaching. However, robots belonging to an institution and deployed in public spaces represent another major use case for robots in the wild. This creates challenges when it comes to learning from humans, since they may receive neither hands-on demonstrations (due to safety concerns) nor constant supervision (due to being away from their supervisor). We develop an algorithmic step after Learning from Demonstration (LfD), in which a robot deployed in the wild with a small number of context-specific skills can use feedback from co-present naïve users to build a more robust action model and self-supervise execution. Our algorithm, Simulation-informed Active In-the-wild Learning (SAIL), enables a robot to learn manipulation skills without direct demonstrations or constant supervision, in which the robot uses a "seed" demonstration provided in-lab and uses a combination of mental simulation and intermittent interaction with co-present people in a public space to learn new valid motion plans and a sensory model of success and failure.

This work was supported by the National Science Foundation (IIS-1564080, IIS-1724157) and Office of Naval Research (N000141612835, N000141612785).

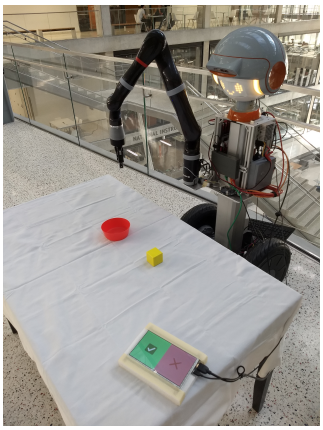


Fig. 1. A robot deployed in a publicly accessible building atrium performs in-the-wild label collection to learn new ways to complete manipulation skills.

Typically, learning algorithms such as reinforcment learning assume that a function mapping sensor/world states to outcomes can be programmed into the robot in advance or obtained from a teacher who is constantly monitoring robot execution, and even in that case, learning is a relatively slow process. Approaches such as policy shaping [1] integrate human feedback to allow the agent to learn faster, but again at the cost of requiring a teacher who can provide multiple demonstrations or constantly monitor and provide feedback on actions. A robot can use LfD to quickly learn new skills, but requires either hands-on demonstrations or a robust mapping from human motions to robot joint configurations, and can only learn policies from sets of consistent demonstrations (for example, always grasping an object in approximately the same position). In addition, for a robot deployed in the real world, it may be desirable to learn many alternative ways of executing a task, each of which requires multiple demonstrations in an LfD framework. In the wild, however, the robot is operating in an

environment of extreme data sparsity: executing trajectories on the physical hardware is time consuming and risky, and even given execution, feedback from a human is rarely available.

We propose a novel approach to addressing these challenges using a 3-step algorithm, in which simulation and unsupervised learning are used to reduce the space of possible trajectory perturbations into a small number of high-information examples. These can be executed in the presence of naïve users in the wild to obtain labels, and build a robust model of task execution without *a priori* knowledge of the execution environment or dedicated supervision. The main contributions of this work are an algorithm for learning a robust execution graph and success model, and an analysis of end-user behavior to provide insights into the frequency and quality of feedback available from naïve users in unconstrained environments. We show that the simulation component is able to reduce the space of potential executions by 75% to those which maximize the change in action and minimize the change in final effect of the behavior, with in-lab success rates of 391 successes to 21 failures and in-the-wild success rates of 95 successes to 68 failures. We also show that the sensor model accurately models success versus failure with a statistically significant difference, and that propagating information between different perturbations of the same action step improves predictions, even for actions without a ground truth label. Finally, we show in a series of 6 45-minute to 1-hour deployments that naïve users are a source of useful information for the robot, with users labeling 163 out of 496 (32.9%) manipulation actions the robot performed in the wild.

II. Related Work

Prior work in HRI has shown that users in public spaces are willing to help a robot [2], although some of these users may be more interested in exploring the robot's capabilities than interacting with it as intended [3]. Other work has examined how to integrate human feedback into an agent's learning process. Some of this work has placed the human in the role of an advisor (see [4] for a unifying framework describing possible human roles), for example to block catastrophic failures [5] or to provide preference information [6]. Other work has modeled and tried to minimize a human supervisor's "surprise" at an agent's actions [7]. Grollman and Billard [8] learn from failed end-user demonstrations by learning to avoid that part of the state space, while Nguyen *et al.* [9] examine how an agent can learn a *set* of optimal plans to satisfy human preferences in which there are trade-offs that are imprecisely specified. In the extreme case, some work has examined how to learn in the total absence of human feedback. For example, Levine *et al.* [10] used several robots working in parallel and without human feedback to train grasping controllers for a wide range of objects. Independent of the human-robot interaction itself, other work has addressed the general problem of modifying learning algorithms to handle uncertainty, such as MDPs [11]–[13]. However, much prior work has assumed that the supervisor will provide some alternative action or policy [6], [7], [14], which may not be possible in the wild, and all of these approaches require large quantities of data, which may not be available in the wild, or hand-specified reward functions, which can be difficult to create. Our approach also does not make any *a priori* assumptions about what constitutes a valid policy, only that it is somehow "near" the original demonstration. Thus our approach takes into account not only the original demonstrator's ideas about how the task might be performed, but will also inherently adapt to any unexpected features of the environment (for example, a more constrained workspace), and any common preferences from the users in a given space (for example, cultural norms).

Researchers have also addressed the problem of how to learn task constraints. Bajcsy *et al.* [15] allow users to correct demonstrations one dimension at a time through physical corrections. Hayes and Scassellati [16] learn task constraints through active learning with a human demonstrator. Cakmak and Thomaz [17] use active learning to enable a robot to query a human teacher about task constraints and goal features at different points in a demonstration.

Other work has used simulations to augment or improve a human demonstration. Večerík *et al.* [18] use demonstrations collected on a real robot arm to guide exploration for a simulated version of the same robot, and use prioritized experience replay to ensure that human demonstration actions are given extra weight during exploration compared to simulated actions. Ugur, Oztop, and Sahin [19] presented a framework for learning object affordances in simulation based on an initial demonstration and executing them on a real robot. Their framework finds clusters of perceptual features to predict affordances and their effects, resulting in a set of affordances that can be integrated with sequential manipulation planning.

Another line of related work has examined how to separate modeled human feedback from the action space. For example, Christiano *et al.* [20] use a two-part model in which both the agent's policy and the mapping from states to rewards are learned by deep networks. They capture human preference by asking the human supervisor to compare between two trajectories. Similarly, Knox and Stone [21] (see also [22]) use human ratings of agent performance to allow an agent to learn directly from human feedback. Akgun and Thomaz [23] simultaneously learn both actions and goals through demonstrations from naïve users, learning both from around ten demonstrations per user. Actor-critic models [24], [25] estimate both the value function (the critic) and search through a parameterized space of policies (the actor). We use the same idea as in these methods of separating the policy evaluation from the policy exploration, however, rather than finding a single optimal policy, the robot is focused on exploring the space of *valid* policies, and maintains a noisy classifier of the human evaluation of the behavior.

III. SAIL Algorithm

As described in Algorithm 1, the SAIL algorithm consists of 3 steps to efficiently learn new ways of completing an action given a single demonstration. After generating perturbed versions of the original demonstration, a simulation

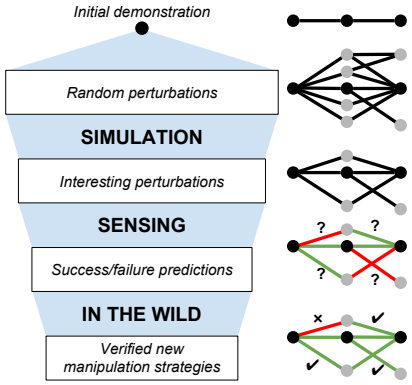


Fig. 2. Our robot uses a combination of simulation (Section III-B), sensing (Section III-C), and spontaneous in-the-wild human interactions (Section III-D) to learn and validate new ways to complete a manipulation task.

step identifies which demonstrations cause a large change in the action space, but with only a small change in the outcome of the behavior. The new candidate demonstrations are added to an execution graph (see Fig. 2). The robot then engages in unsupervised execution of the trajectories by traversing edges in the execution graph while monitoring low-level sensor data. As trajectories are executed, the algorithm predicts the success or failure of each trajectory based on prior and current sensory data and obtains labels from humans in the environment when available, propagating the success/failure information to other trajectories using the previously-collected unsupervised data.

Algorithm 1 SAIL Algorithm

Given: Task list $t_{1...num\_tasks}$, tuning parameter α
$D \leftarrow$ CollectDemonstration()
for $n = 0$ to N **do**
  $D'_n \leftarrow$ PerturbDemonstration(D)
  $d'_{a_n}, d'_{e_n} \leftarrow$ Simulate(D'_n)
  $\psi_n = d'_{a_n}\alpha + (1 - d'_{e_n})(1 - \alpha)$
end for
$execution\_graph \leftarrow$ Top$\psi(D', 0.25)$
$node \leftarrow$ StartNode(t_0)
while true **do**
  $edge \leftarrow$
$\arg\min(|\text{Prob}(fail, edge) - \text{Prob}(success, edge)|)$
  $result, sensory\_features \leftarrow$ ExecuteEdge$(edge)$
  if $result = error$ **then**
    $label \leftarrow error$
  else if human present **then**
    $label \leftarrow$ CollectFeedback()
  else
    $label \leftarrow unknown$
  end if
  UpdateProbs$(task, edge, sensory_features, label)$
  $node \leftarrow$ Traverse$(node, edge)$
  if $node$ is terminal **then**
    $task \leftarrow$ NextTask$(task)$
    $node \leftarrow$ StartNode$(task)$
  end if
end while

A. Preliminaries

The algorithm takes as input a single demonstration, which consists of a sequence of K keyframes. Each keyframe is represented by an end effector pose, gripper status (open or closed), and object $m \in M$ defining the reference frame of the keyframe. The object may be the base of the robot, or any other object in the workspace that the robot is otherwise aware of (such as those detected by a perception algorithm). The demonstration is provided by an expert to provide a good initial set of keyframes on which to base further exploration.

An "execution" of the demonstration consists of the robot planning and moving to each of the poses in the reference frame of the object. It produces a graph of keyframes, where an edge in the graph connects two keyframes. To execute the a skill, the robot follows a path through the graph, moving its end effector to the position specified by each keyframe in the path. While moving, the robot records from the sensors (e.g., the effort from one joint) and associates the data with the edge between the two keyframes. The data from all edges in a given step are combined, allowing the robot to infer whether an edge corresponds to success or failure even if only unsupervised executions of that edge have occurred.

We assume that a given skill has a fixed number of steps, although the approach could be extended to include adding or removing keyframes. We do not perturb a demonstrations' first keyframe, since it provides a common starting point and can represent a static out-of-the-way pose for perception. This work is also restricted to skills where the effect of a skill is encapsulated by the changing position of objects. Some tasks, such as pushing a button, may not be defined in this way, and require constructing an effect definition that captures other signals. Finally, this approach (described in Section III-C) makes the assumption that Gaussian distributions can be used to model the sensor data.

B. Discovering New Keyframes using Simulation

Beginning with a skill demonstration collected from a human expert, we perturb the demonstration to expand an execution graph (see Fig. 2) that represents other possible ways of completing the skill. Since we are seeking new ways of completing the same task, rather than discovering entirely new skills, we seek perturbations that use different actions (in our case, using a demonstration in which one or more keyframes may be different from the original demonstration) to achieve the same end result.

For an initial K-keyframe demonstration, we randomly select between 1 and $K - 1$ keyframes to perturb. The selected keyframes are perturbed along one of their coordinate frame basis vectors, chosen at random. We use the current frame's basis vectors (X, Y, and Z axes) as the directions for perturbations because the coordinate frame's orientation holds important knowledge about the world. For example, in our tasks, the Z-axis is coincident with gravity and the other

two axes are parallel to the table, making them interesting directions for exploration. The keyframes are either rotated about the basis vector by a random amount in the range $[-\pi/2, \pi/2]$, or translated a random amount along that vector in the range $[-0.2\text{m}, 0.2\text{m}]$.

We can generate any number of perturbations of the original demonstration and perform them in simulation, but not all of them will be good candidates to replay on the robot. To evaluate how valuable a perturbed demonstration is for further data collection, we introduce the value score metric, ψ. Demonstrations with a high ψ represent a "different way" of completing the same task as the original demonstration.

The value score is calculated as follows. For each perturbed demonstration, we first use Eq. (1) to calculate the *action distance* (d_a): how much the new set of perturbed keyframes differ from those in the original demonstration, summed over all K keyframes in the demonstration. In this equation, $\boldsymbol{x}_k$ is the robot end effector's XYZ position at keyframe k with respect to that keyframe's coordinate frame. The coordinate frame may be co-located with the robot base or one of the objects in the scene, depending on the original demonstration. θ_k is the smallest angle of rotation (in radians) between the original and perturbed keyframe end effector poses.

$$d_a = \sum_{k=1}^{K} \left\| \boldsymbol{x}_{k_{perturbed}} - \boldsymbol{x}_{k_{original}} \right\|^2 + \frac{\theta_k}{4} \tag{1}$$

Similarly, the *effect distance* (d_e) for each demonstration, given by Eq. (2), is how much the final positions of the M objects in the scene differ from their final positions under the original demonstration.

$$d_e = \sum_{m}^{M} \left\| \boldsymbol{x}_{m_{perturbed}} - \boldsymbol{x}_{m_{original}} \right\|^2 \tag{2}$$

Because the action and effect distances depend on the task and arm geometry, they will have varying magnitudes, so we normalize them, defining $d'_a = d_a / \max_{n=1\ldots N} d_{a_n}$ and $d'_e = d_e / \max_{n=1\ldots N} d_{e_n}$. Finally, we calculate the value score ψ for each perturbed demonstration according to Eq. (3).

$$\psi_n = d'_{a_n} \alpha + (1 - d'_{e_n})(1 - \alpha) \tag{3}$$

This equation assigns high value score to perturbed demonstrations for which the action distance d'_a is high and the effect distance d'_e is low. The parameter $\alpha \in [0, 1]$ is used to tune the preference between maximizing the action distance and minimizing effect distance. For example, $\alpha = 1$ will assign the highest value score to perturbations with the maximum action distance, regardless of the effect.

Once ψ has been calculated for all perturbed demonstrations, we select the top 25% highest-scoring demonstrations[1] and add them to the execution graph for real-world evaluation on the robot. Since some keyframes remain the same as the original demonstration, the manipulation graph will contain branches representing the different perturbations.

[1]Using our perturbation method, the top 24.8% of perturbed demonstrations account for 50% of the total variation in value score. This was calculated by conducting 1000 perturbations, simulating them, and calculating their scores.

C. Sensor Models of Action Completion

The sensor model allows the robot to self-supervise during execution and to more accurately infer what perturbations correspond with successful task execution. As discussed in Section III-A, the sensor model is built during execution, and is associated with a specific step of the task, incorporating information from all edges corresponding to that step.

To control for different execution times for different perturbations, time is normalized to the interval $[0, 1]$ and the sensor data resampled so that each segment contains the same number of time points. At each time point for each feature, the sensor model maintains a Gaussian distribution on the samples that have been labeled as "success" and the samples that have been labeled "failure". The probability of a sample belonging to the "success" or "failure" class (function PROB in Algorithm 1) is the mean probability of the sensor data belonging to the relevant classifier at each time for each feature. That is, if S is the set of sensor data, t is the time point, $s_t \in S$ is the current level of the sensor data at time t, Φ is the cumulative distribution function of the normal distribution, and $\mu_{success,t}$ and $\sigma_{success,t}$ are the mean and standard deviation of the samples at time t labeled as success, then the probability that a sample belongs to the "success" model is:

$$\frac{1}{|S|} \sum_{s \in S} \left(\frac{1}{|t|} \sum_{t} \Phi((s_t - \mu_{success,t})/\sigma_{success,t}) \right) \tag{4}$$

The probability that a sample belongs to the "failure" model is calculated in the same way; note that these probabilities need not sum to one, since the probabilities are calculated independently. For a maximally ambiguous sample, the probability of belonging to "success" or "failure" are be equal, thus in a minimally informative model, all samples are be ambiguous. These probabilities can also be calculated for unlabeled data, and used to determine which edges are most informative to request a label for, as described in the next section. They are updated in the function UPDATEPROBS in Algorithm 1. The key insight of this work is that $\mu_{success,t}$ and $\sigma_{success,t}$ are calculated across *all edges belonging to the same step of the task*, so that when a label is obtained for one edge, the information is propagated to all edges. This allows the robot to infer the success or failure of unlabeled edges, as long as some edge in that step has been labeled.

D. In-The-Wild Labeling

The final step in the SAIL algorithm is to obtain labeled samples from naïve users in the environment. In the deployment environment, the robot collects unlabeled data when no human is present (or when a co-present human fails to label a segment), and collects labeled data when labels are available. There are four possible labels: *success*, when a step is completed correctly, *failure*, when a step is executed by the robot but does not correctly progress the task, *error*, when

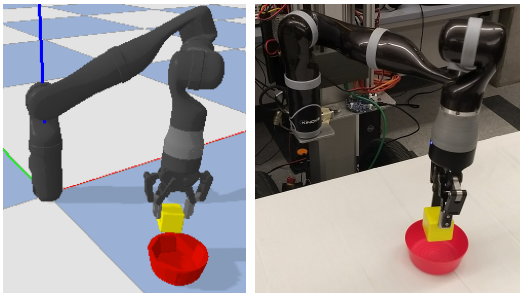


Fig. 3. Left: The simulated robot arm completes the *pick and place* task. Right: a perturbed version of the same keyframe (rotated about the vertical axis) being replayed on the real robot arm.

there is a problem with the robot's motion plan, and *unknown*, when a label was not provided for an execution. The labels are associated with a specific set of sensor data from a single pass through an edge; the sensor model combines data across edges corresponding to the same step, allowing the robot to infer whether edges with only *unknown* labels primarily result in success or failure. We compare two approaches to choosing a path through the execution graph when making queries to humans in the wild: in the first approach, the robot greedily chooses a path so that it minimizes the difference between the probability of success and the probability of failure for the segment, based on the average sensor model probabilities associated with those labels for all previous executions of that edge. In the second approach, the robot chooses the shortest error-free path to the least-sampled edge in the graph.

IV. System Implementation

We evaluated the system relative to two manipulation skills. The first is a *pick and place* skill, where the robot picks up a yellow block and places it in a red bowl. The second skill is a *pour* skill, where the robot picks up the bowl by its rim and dumps the block onto the table. These skills are good candidates for our study because a) they involve multiple steps, b) both translation and rotation of the robot's end effector are important, especially on the *pour* skill, and c) successfully completing one skill puts the environment in the starting state for the next skill, so the skills can be alternated repeatedly without human intervention. The skills were created by recording a kinesthetic demonstration (as in [26]), where an expert user physically moved the robot's gripper through the steps of the skill and recorded keyframes along the way.

After generating perturbations as described in Section III-B, We replayed each of the $N = 40$ generated perturbations in the simulator and calculated their value score as described in Section III. We set $\alpha = 0.3$ for both skill in this study, determined empirically based on our choice of tasks. The top 10 (25%) perturbations, capturing around 50% of the variation in value score, are added to the execution graph (see Fig. 2).

Both evaluations were conducted using a custom mobile manipulation robot, equipped with a Kinova Jaco 7 degree-of-freedom arm, a Robotiq 85 2-finger gripper, and an articulated neck joint. The platform also has an LED array in its head which displays a simple face, as well as LED ear lights and a speaker for speech using the Amazon Polly text-to-speech engine. The robot collects sensory input using an Intel RealSense D400 depth camera, MiniDSP directional microphone array, and the arm's built-in end-effector effort measurements. To simulate perturbed versions of the actions, we created a testbed using the Bullet simulator (Fig. 3). The models for the manipulation objects were created using Blender and V-HACD[2], and were assigned mass of 0.1kg and friction of $\mu 1 = 1.0, \mu 2 = 0.001$. The simulated arm and gripper followed recorded keyframes using inverse kinematics (for the arm) and simulated constraints (for the gripper linkage).

The low-level features used in this work were gripper state, joint effort, sparse optical flow, audio energy in 5 frequency bands, audio intensity, spectral flatness, and total motion. We did not include the end effector position or object positions because of noise in the tracking and motion planning, although these might improve performance for some tasks. The sensor data was resampled to contain 50 samples per feature in each segment, normalizing over time. We added two features to the planning framework to ensure that the tasks could be successfully replayed without human intervention in a real-world environment. By combining object-relative keyframes with a perception system, the robot could continue to execute skills as objects were moved around the workspace. The perception system used the robot's on-board depth camera and point cloud processing algorithms to detect object positions and colors. We also defined a "recovery motion" for the robot. In the event of a planning failure, the robot moved to a recovery pose, dropped what it was holding, and began again.

V. Evaluation

We evaluated each step of the algorithm, using a combination of expert labeling and in-the-wild testing. For the in-lab evaluation, two execution graphs for each task were fully labeled by an expert. To do this, the robot first built up an unlabeled model of task execution by sampling the least-sampled edges in the graph until every edge was either traversed once or determined to be unreachable (reaching it required traversing an edge that resulted in a motion planning error), or until at least ten samples were obtained for every step in the task. This unlabeled model was then used to bootstrap two rounds of expert labeling, one using the information-based sampling approach to sampling and one using the coverage-based approach. In the coverage-based approach, expert labeling continued until all edges reachable by an error-free path were labeled, and in the information-based approach, the expert provided labels until the model selected the same set of keyframes three times in a row.

We evaluated the full system in the wild to show that the robot is able to learn a useful model of success and failure, as well as characterizing the frequency, duration, and type of interactions that occur with users. To collect human feedback, we placed a touch screen on the table in front of

[2]https://github.com/kmammou/v-hacd

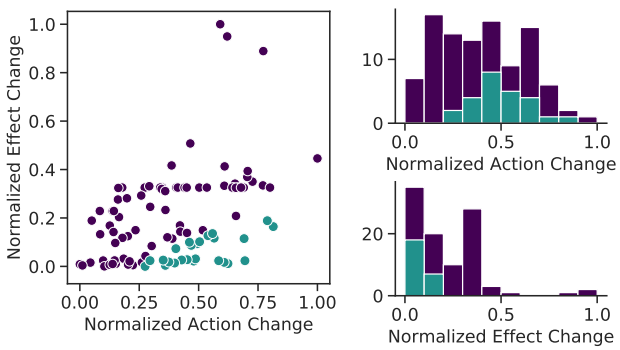
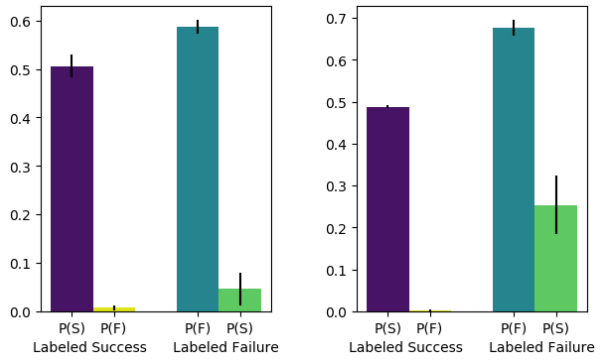


Fig. 4. 100 simulated perturbations of the *block dropoff* task, showing a weak correlation between increasing action perturbation and the size of the resulting effect (both normalized). The top 25% of samples (ranked by value score, see Section III-B) are highlighted in blue/light.



(a) Naïve Users (b) Expert User

Fig. 5. Sensor model's estimation of probabilities for labeled edges. The majority of paths resulted in successful executions in the lab, resulting in a noisier failure classifier in-lab. Although values are in [0,1] the total probability does not sum to 1 because the classifiers are trained separately.

the robot. The screen displayed a green "success" button and a red "failure" button (see Fig. 1). Person detection was not a core contribution of this work so we used a human-in-the-loop approach: an observer in a nearby study area labeled the number of available interaction partners using a laptop.

The robot executed manipulation tasks in the atrium of a publicly-available building between the hours of 10:00 and 18:00. The robot was placed behind a table, but was not accompanied by a human "handler" or any instructional material, rather, all interactions were spontaneous, and initiated upon interest from a passer-by. Whether or not it was interacting with a human, the robot continuously selected manipulation actions to perform using one of the two selection strategies described above, alternating between the "pick" and "pour" tasks as it completed tasks. If the robot motion planner encountered an error or was unable to find a valid plan after 10 attempts, the robot returned its arm to a known joint pose and selected a new set of keyframes. Data collection was completed in 6 sessions, 3 for each of the two action selection conditions (information-based and coverage-based).

When a passer-by approached, the robot elicited feedback from them. After finishing its current manipulation action, the robot asked for the participant's help ("I'm learning how to use this stuff. If you could help me out, that would be awesome") and provided instructions. Participants were instructed to press the check mark button if the action "looked good" or "looked right," and otherwise to press the X button. The robot and the participant then took turns acting and providing feedback, respectively, until the participant chose to leave. Manipulation actions were chosen according to the currently active selection strategy, as described in Section III. During this time, the robot spoke after each motion ("OK?" "And?"), when feedback was supplied ("Great!", "Oh no!"), at the end of a successful task ("Yay!"), and if an error occurred ("I ran into an error, I'm going to start again from the beginning"). Collecting labels in this manner means that every keyframe executed resulted in one labeled data point for that edge in the execution graph. The label was either "success", "failure", "unknown" (motion planning succeeded, but no human feedback was available), or "error" (motion planning failure).

VI. Results

Fig. 4 illustrates the resulting behavior when a keyframe demonstration is perturbed 100 times in our simulator. The data, while noisy, points to a moderate correlation (Spearman's $\rho = 0.44$, $p = 3.4 \times 10^{-6}$) between the size of an action perturbation and the resulting change in effect. Measuring perturbations via simulation using our approach is subject to large uncertainty, but is still more informative than making every possible random perturbation and trying them all on the robot. This confirms our choice of the simulator as a good first step for our filtering process, and the filtering is further improved by the use of the value score.

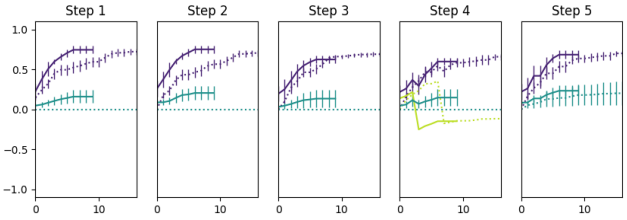
Fig. 5 shows the accuracy of the sensor model in representing success versus failure of edges in the execution graph. For expert labeling, the average probability of a sensor sample that was labeled "success" belonging to the success classifier (true positive) was 0.488 ($SD = 0.041$), while the probability a sensor sample labeled "success" belonging to the failure classifier (false negative) was 0.00184 ($SD = 0.014$). This was a significant difference ($t(104) = 116.9, p < .0001$; paired t-test). There was also a significant difference between the probability of a "failure" sample belonging to the failure distribution (true negative; $M = 0.676, SD = 0.32$) and the probability of the sample belonging to the success distribution (false positive; $M = 0.254, SD = 0.19$; $t(20) = 4.65, p < 0.0005$; paired t-test). This difference also held in the sensor data collected in the wild, with the probability of a sensor sample labeled "success" belonging to the success distribution ($M = 0.506, SD = 0.19$) significantly higher than the probability of that sensor sample belonging to the failure distribution ($M = 0.0456, SD = 0.11$; $t(64) = 20.34, p < .0001$; paired t-test). Similarly, the probability of a sensor sampled labeled as "failure" in the wild belonging to the failure distribution ($M = 0.587, SD = 0.25$) was higher than the probability of it belonging to the success distribution($M = 0.00713, SD = 0.045$; $t(52) = 14.02, p < 0.0001$; paired t-test).

Fig. 6 shows how the difference between the success probability and failure probability, as calculated in Equation 4, evolves as new labels are added to the model. This measure varies from -1 (very certain to be failure) to 1 (very certain to be success). By plotting this value for edges with different ground truth labels, we can show that for edges that are known to result in successful execution, this value evolves towards 1.0, for edges that are known to result in failure this value evolves towards -1.0, and for edges that are ambiguous (may result in success or failure), the value stays close to 0. When this value more quickly goes towards 1 or -1, then the system is learning faster; thus Figure 6 also shows that using the information-based approach to collect labels allows the robot to more quickly distinguish between successful and failed executions.

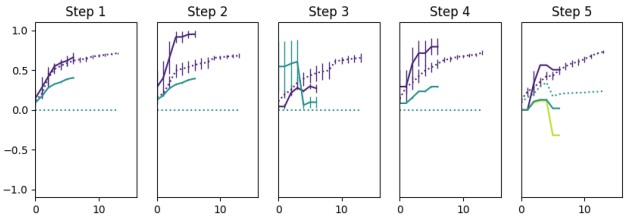
Of the 496 manipulation actions the robot performed in the wild, 163 of them (32.9%) were labeled by human participants. For the picking task, naïve users labeled 45 successes and 28 failures, while 133 actions were unlabeled and 50 resulted in an error condition (typically a planning failure). The expert labeled 185 successes and 4 failures in the same task, with 15 errors. For the pouring task, naïve users labeled 50 successes and 40 failures, with 200 unlabeled actions and 25 errors. The expert user labeled 206 successes and 17 errors in the pouring task, with 27 errors. On average, one label was provided for every 120 seconds the robot was operating in the wild. Other than the labeling of the number of available interaction partners, the robot operated with minimal intervention from the experimenters, with interventions needed every 32 minutes on average (10 total). In seven of those cases, the experimenter needed to reset objects that were pushed/poured out of reach or off the table; in one case the objects ended up too close together for the perception pipeline to detect them; in one case the robot tipped the bowl sideways, making both detection and grasping impossible, and in the last case, a participant walking away without giving feedback froze the control code.

The learning pipeline was evaluated in the wild in 2 one-hour pilot sessions and four 45-minute experimental sessions. The pilot sessions were identical to the main experiment sessions except that the robot chose its path through the execution graph randomly if no human was present at the beginning of the task and asked the user to wait if they approached the robot while it was executing a demonstration. In these pilot sessions, it was found that the majority of people left after 1-2 keyframes (20-25% of a demonstration). This behavior can be observed in Fig. 7. The algorithm was then changed to always choose an informative (or coverage-increasing) path and ask for a label whenever a person was present to take maximal advantage of the small amount of human feedback available. In the experimental sessions, most people still stayed for only a few keyframes, but the robot was able to take advantage of the labels they provided. The results in Figure 5 include all six sessions.

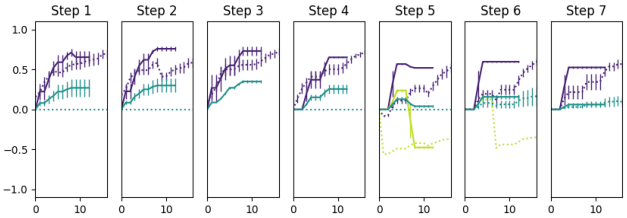
We consider "robustness" to be based on the number of new valid execution strategies found. For the first perturbation of the pick task, the algorithm initially added 36 edges to



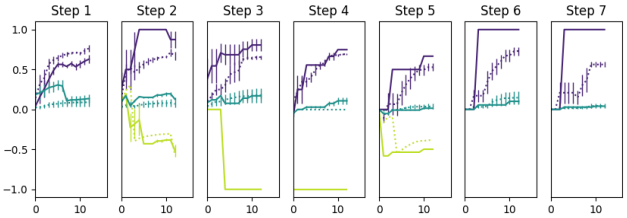
(a) Pickup Task Perturbation 1



(b) Pickup Task Perturbation 2



(c) Pouring Task Perturbation 1



(d) Pouring Task Perturbation 2

Fig. 6. The mean difference between success and failure probabilities of all edges belonging to a step, for edges with ground truth labels of successful (purple/dark), unknown (blue/medium; equal numbers of expert labels of success and failure), and failure (green/light). That the majority of paths returned by the simulator resulted in successful executions in the lab. Change over time for the information-based approach is shown in solid line, while the coverage-based approach is shown with a dashed line. Vertical axis is the difference in probabilities; horizontal axis is the number of labels obtained for any edge in that step.

the original 5-edge graph, 30 of which had equal or more success than failure in the expert labeling, resulting in 49 possible paths through the graph, or 49 different ways of executing the task. For the second perturbation, the algorithm added 29 edges, of which 23 were successful, resulting in 36 ways to execute the task. For the first perturbation of the pour task, the algorithm initially added 56 edges, of which 50 were successful, resulting in 70 possible executions, and for the second perturbation, which had the lowest success rate, the algorithm initially added 57 edges, of which 26 were successful, resulting in 27 ways of executing the task. Given that in the initial demonstration, the robot only had one possible way to execute the task, these numbers represent

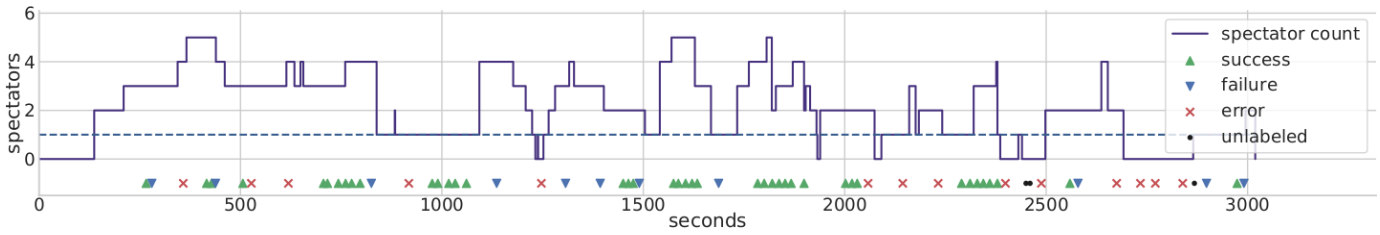


Fig. 7. The labels collected and number of participants during one of the in-the-wild sessions in our study (session 4, coverage edge selection method).

a considerable improvement in the options the robot has to choose from for executing the task: that is, if one approach is blocked for some reason, the robot retains many more options than it initially had for executing the task.

VII. Discussion and Conclusion

Our work shows that a keyframe demonstration can be successfully modified by using simulated perturbation followed by real-world evaluation, even with naïve users. The SAIL algorithm uses three steps to identify the most promising alternate versions of the demonstration using unsupervised techniques, infer the success or failure of unlabeled executions from a small set of labeled executions, and validate this prediction using explicit feedback from human-robot interaction. The simulator is able to significantly reduce the space of perturbations to test, and the majority of the resulting perturbations resulted in successful execution, especially in the lab environment. Furthermore, low-level sensor data can be used to distinguish between successful and failed executions (Fig. 5) and that propagating the information across different ways of executing a task step can help the robot to choose informative behaviors to ask about (Fig. 6), although when many labels are available, it may be better to fall back to asking about every edge. Ultimately, in this work the human is an important part of the learning process, where the robot makes efficient use of feedback by only asking about actions that have already been vetted by previous steps.

Qualitatively, we observed that the perturbations produced by the simulation algorithm varied, from rotations in the end-effector position when dropping the block in the box, to one variation in the pour task where the robot pushes the bowl to one side and then picks it up from the opposite lip from the grasp pose in the original demonstration. We observed many participants filming the robot or taking pictures, and approaching the robot in groups. However, many more people walked by the robot without stopping, briefly glanced at the robot, or stopped for only a few seconds. Participants sometimes picked up and moved the objects on the table, placing them where the robot could not reach them or attempting to put them in the robot's gripper. In addition, we observed that participants often continued to provide feedback until the robot either failed a step (as determined by the participant) or encountered a motion planning error, after which some or all of the participants left.

This work represents a step towards the goal of an always-on mobile manipulator learning from naïve users in the wild. Although each component of this model could be improved (e.g., incorporating task-specific information to generate nonrandom perturbations), we have shown that this multi-step approach can be used to efficiently learn new task execution approaches. We show that there is a strong separation between the low-level sensor data for "success" and "failure", suggesting that other machine learning techniques can also be expected to generate accurate classifiers. However, note that although the Gaussian approach used in this work requires strong assumptions of normality, it is less sensitive to the so-called "curse of dimensionality" than some other approaches; we qualitatively observed that many of the features were uninformative, but that lack of information is captured in the noise model and ignored by the system. Relaxing the normality assumption or allowing multimodal distributions could improve performance but would require more samples.

Future work could address how to integrate more information between steps, for example, using execution monitoring to improve the fidelity of the simulator. Other work might study how to more tightly integrate this approach with other learning algorithms. Another direction is to more effectively engage passers-by in the interaction, especially those who look at the robot and are interested, but only pause briefly without stopping. The current approach is sensitive to the quality of the initial demonstration; this work could be extended by learning an initial demonstration from multiple examples, after which the SAIL algorithm could be used. Our approach assumes that the initial demonstration includes information about the coordinate frame in which the action is taken (that is, when reaching for the block, that the action is in the blocks coordinate frame). In practice, this information is easily provided by the demonstrator during the initial keyframe demonstration, and provides an adequate initial basis for perturbing keyframes. Future work could explore how to simulate the robot actions even more quickly and enable more random perturbations or exploring perturbations in configuration space.

In conclusion, the SAIL algorithm bridges LfD and planning to enables a robot to learn key constraints and alternative approaches to executing an action of interest. We show that it is possible to learn new approaches to manipulation actions from naïve users in the wild, using a combination of simulation and low-level sensor data to identify promising candidate actions, distinguish between successful and unsuccessful motions in a multi-step task, and to choose informative candidate actions to execute with human users.

References

[1] 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 26*, C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, Eds., Curran Associates, Inc., 2013, pp. 2625–2633. [Online]. Available: http://papers.nips.cc/paper/5187-policy-shaping-integrating-human-feedback-with-reinforcement-learning.pdf.

[2] A. Weiss, J. Igelsböck, M. Tscheligi, A. Bauer, K. Kühnlenz, D. Wollherr, and M. Buss, "Robots Asking for Directions: The Willingness of Passers-by to Support Robots," in *Proceedings of the 5th ACM/IEEE International Conference on Human-robot Interaction*, ser. HRI '10, Piscataway, NJ, USA: IEEE Press, 2010, pp. 23–30, ISBN: 978-1-4244-4893-7. [Online]. Available: http://dl.acm.org/citation.cfm?id=1734454.1734468.

[3] S Andrist, D Bohus, Z Yu, and E Horvitz, "Are you messing with me? Querying about the sincerity of interactions in the open world," in *2016 11th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, Apr. 2016, pp. 409–410. DOI: 10.1109/HRI.2016.7451780.

[4] F. Benavent and B. Zanuttini, "An Experimental Study of Advice in Sequential Decision-Making under Uncertainty," in *32nd AAAI Conference on Artificial Intelligence (AAAI-18)*, New Orleans, USA, Feb. 2018. [Online]. Available: https://hal.archives-ouvertes.fr/hal-01646200.

[5] W. Saunders, G. Sastry, A. Stuhlmüller, and O. Evans, *Trial without Error: Towards Safe Reinforcement Learning via Human Intervention*, 2018. [Online]. Available: https://dl.acm.org/citation.cfm?id=3237383.3238074.

[6] P. Alizadeh, Y. Chevaleyre, and F. Levy, "Advantage based value iteration for Markov decision processes with unknown rewards," in *2016 International Joint Conference on Neural Networks (IJCNN)*, IEEE, Jul. 2016, pp. 3837–3844, ISBN: 978-1-5090-0620-5. DOI: 10.1109/IJCNN.2016.7727695. [Online]. Available: http://ieeexplore.ieee.org/document/7727695/.

[7] K. Amin, N. Jiang, and S. Singh, "Repeated Inverse Reinforcement Learning," in *Advances in Neural Information Processing Systems (NIPS)*, Long Beach, CA, 2017, pp. 1815–1824. [Online]. Available: http://papers.nips.cc/paper/6778-repeated-inverse-reinforcement-learning.

[8] D. H. Grollman and A. Billard, "Donut as I do: Learning from failed demonstrations," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, IEEE, 2011, pp. 3804–3809.

[9] T. A. Nguyen, M. Do, A. E. Gerevini, I. Serina, B. Srivastava, and S. Kambhampati, "Generating diverse plans to handle unknown and partially known user preferences," *Artificial Intelligence*, vol. 190, pp. 1–31, Oct. 2012, ISSN: 0004-3702. DOI: 10.1016/J.ARTINT.2012.05.005. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0004370212000707.

[10] S. Levine, P. Pastor, A. Krizhevsky, J. Ibarz, and D. Quillen, "Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection," *The International Journal of Robotics Research*, p. 027 836 491 771 031, Jun. 2017, ISSN: 0278-3649. DOI: 10.1177/0278364917710318. [Online]. Available: http://journals.sagepub.com/doi/10.1177/0278364917710318.

[11] A. Ahmed, P. Varakantham, M. Lowalekar, Y. Adulyasak, and P. Jaillet, "Sampling Based Approaches for Minimizing Regret in Uncertain Markov Decision Processes (MDPs)," *Journal of Artificial Intelligence Research*, vol. 59, pp. 229–264, Jul. 2017, ISSN: 1076-9757. DOI: 10.1613/jair.5242. [Online]. Available: https://jair.org/index.php/jair/article/view/11066.

[12] K. Regan and C. Boutilier, "Robust Policy Computation in Reward-Uncertain MDPs Using Nondominated Policies," in *Twenty-Fourth AAAI Conference on Artificial Intelligence (AAAI-10)*, Atlanta, Georgia, USA, 2010, pp. 1127–1133. [Online]. Available: www.aaai.org.

[13] V. Freire da Silva and A. H. Reali Costa, "A Geometric Approach to Find Nondominated Policies to Imprecise Reward MDPs," in, Springer, Berlin, Heidelberg, 2011, pp. 439–454. DOI: 10.1007/978-3-642-23780-5_38. [Online]. Available: http://link.springer.com/10.1007/978-3-642-23780-5{_}38.

[14] R. Lioutikov, G. Neumann, G. Maeda, and J. Peters, "Learning movement primitive libraries through probabilistic segmentation," *The International Journal of Robotics Research*, vol. 36, no. 8, pp. 879–894, Jul. 2017, ISSN: 0278-3649. DOI: 10.1177/0278364917713116. [Online]. Available: http://journals.sagepub.com/doi/10.1177/0278364917713116.

[15] 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 - HRI '18*, New York, New York, USA: ACM Press, 2018, pp. 141–149, ISBN: 9781450349536. DOI: 10.1145/3171221.3171267. [Online]. Available: http://dl.acm.org/citation.cfm?doid=3171221.3171267.

[16] B. Hayes and B. Scassellati, "Discovering task constraints through observation and active learning," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, Sep. 2014, pp. 4442–4449, ISBN: 978-1-4799-6934-0. DOI: 10.1109/IROS.2014.6943191. [Online]. Available: http://ieeexplore.ieee.org/document/6943191/.

[17] M. Cakmak and A. L. Thomaz, "Designing robot learners that ask good questions," in *Proceedings of the seventh annual ACM/IEEE international conference on Human-Robot Interaction - HRI '12*, New York,

New York, USA: ACM Press, 2012, p. 17, ISBN: 9781450310635. DOI: 10.1145/2157689.2157693. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2157689.2157693.

[18] 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*, Jul. 2017. arXiv: 1707.08817. [Online]. Available: http://arxiv.org/abs/1707.08817.

[19] E. Ugur, E. Oztop, and E. Sahin, "Goal emulation and planning in perceptual space using learned affordances," *Robotics and Autonomous Systems*, vol. 59, no. 7-8, pp. 580–595, 2011, ISSN: 09218890. DOI: 10.1016/j.robot.2011.04.005. [Online]. Available: https://ac.els-cdn.com/S0921889011000741/1-s2.0-S0921889011000741-main.pdf?{_}tid=08859f92-0150-11e8-b0a8-00000aacb35d{\&}acdnat=1516830577{_}7f407967ff97a91cd85085040bf5542f.

[20] P. Christiano, J. Leike, T. B. Brown, M. Martic, S. Legg, and D. Amodei, "Deep reinforcement learning from human preferences," in *Advances in Neural Information Processing Systems (NIPS)*, Long Beach, CA, 2017. arXiv: 1706.03741. [Online]. Available: https://papers.nips.cc/paper/7017-deep-reinforcement-learning-from-human-preferences.pdfhttp://arxiv.org/abs/1706.03741.

[21] W. B. Knox and P. Stone, "Reinforcement learning from simultaneous human and MDP reward," *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems-Volume 1*, pp. 475–482, 2012. [Online]. Available: https://dl.acm.org/citation.cfm?id=2343644.

[22] W. B. Knox and P. Stone, "Interactively shaping agents via human reinforcement," in *Proceedings of the fifth international conference on Knowledge capture - K-CAP '09*, New York, New York, USA: ACM Press, 2009, p. 9, ISBN: 9781605586588. DOI: 10.1145/1597735.1597738. [Online]. Available: http://portal.acm.org/citation.cfm?doid=1597735.1597738.

[23] B. Akgun and A. Thomaz, "Simultaneously learning actions and goals from demonstration," *Autonomous Robots*, vol. 40, no. 2, pp. 211–227, Feb. 2016, ISSN: 0929-5593. DOI: 10.1007/s10514-015-9448-x. [Online]. Available: http://link.springer.com/10.1007/s10514-015-9448-x.

[24] S. Bhatnagar, R. S. Sutton, M. Ghavamzadeh, and M. Lee, "Natural actorcritic algorithms," *Automatica*, vol. 45, no. 11, pp. 2471–2482, Nov. 2009, ISSN: 0005-1098. DOI: 10.1016/J.AUTOMATICA.2009.07.008. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0005109809003549.

[25] V. R. Konda and J. N. Tsitsiklis, "Actor-Critic Algorithms," in *Advances in Neural Information Processing Systems (NIPS)*. [Online]. Available: http://papers.nips.cc/paper/1786-actor-critic-algorithms.pdf.

[26] B. Akgun, M. Cakmak, J. W. Yoo, and A. L. Thomaz, "Trajectories and keyframes for kinesthetic teaching: A human-robot interaction perspective," in *2012 7th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, Mar. 2012, pp. 391–398. DOI: 10.1145/2157689.2157815.