Learning Robot Motion Control from Demonstration and Human Advice

Brenna D. Argall, Brett Browning and Manuela Veloso

***Abstract*— As robots become more commonplace within society, the need for tools that enable non-robotics-experts to develop control algorithms, or *policies*, will increase. Learning from Demonstration (LfD) offers one promising approach, where the robot learns a policy from teacher task executions. In this work we present an algorithm that incorporates human teacher feedback to enable policy improvement from learner experience within an LfD framework. We present two implementations of this algorithm, that differ in the sort of teacher feedback they provide. In the first implementation, called *Binary Critiquing (BC)*, the teacher provides a *binary indication* that highlights poorly performing portions of the execution. In the second implementation, called *Advice-Operator Policy Improvement (A-OPI)*, the teacher provides a *correction* on poorly performing portions of the student execution. Most notably, these corrections are continuous-valued and appropriate for low level motion control action spaces. The algorithms are applied to validation domains, one simulated and one a Segway RMP platform. For both, policy performance is found to improve with teacher feedback. Specifically, with BC learner execution success and efficiency come to exceed teacher performance. With A-OPI task success and accuracy are shown to be similar or superior to the typical LfD approach of correcting behavior through more teacher demonstrations.**

I. INTRODUCTION

As robots become more prevalent within general society, the need for programming techniques that are accessible to non-experts increases. To develop a control *policy*, or mapping from world observation to action selection, traditional approaches first model world dynamics and then derive the policy mathematically. Though theoretically well-founded, these approaches depend heavily on the accuracy of the world model, which requires considerable expertise to develop. Most other approaches to policy development are similarly restricted in use to robotics-experts.

One potential exception is policy development through *Learning from Demonstration (LfD)*, e.g. [5], [10]. In this paradigm, a teacher first demonstrates a desired behavior to the robot. The robot then generalizes from these examples to derive a policy. Demonstration has the attractive feature of being an intuitive communication medium for humans, who already use demonstration to teach other humans. Since it does not require expert mathematical knowledge of the system dynamics, demonstration also opens policy development to non-robotics-experts.

A desirable feature in any learning system is the ability to improve a policy based upon learner experience. Though not an explicit part of classical LfD, many LfD systems do take policy improvement steps. The most common approach is to add more teacher demonstration data in problem areas of the state space, and then re-derive the policy [7], [8].

In this work we present an algorithm for policy improvement within an LfD framework. Within this algorithm, a teacher is employed not only for demonstration, but also for later evaluations of the learner executions. Furthermore, the techniques we present are appropriate for policy improvement within low level motion control domains. We discuss two implementations of this algorithm, that differ in the type of human teacher feedback provided.

In the first implementation, named *Binary Critiquing (BC)*, the human teacher flags poor performing areas of the learner executions. The learner uses this information to modify its policy, by penalizing the underlying demonstration data that supported the flagged areas. We argue that this sort of feedback is well-suited for human teachers. That is, humans are generally good at assigning credit to a given performance, but have less intuition about assigning credit to the underlying algorithm. Furthermore, under this formulation, a robot mechanism does not need to be performable or understood by the human teacher to receive credit.

In the second implementation, named *Advice-Operator Policy Improvement* (A-OPI), the human teacher provides *corrections* on the learner executions. This is in contrast to BC, in which poor performance is flagged, but the correct action to take instead is not indicated. The learner uses these corrections to *synthesize* new data based on its own executions, and then updates its policy. This alternative source for data, that does not derive from teacher demonstrations, is a key novel feature of the A-OPI algorithm.

We validate these approaches to providing human teacher feedback with simulated and real world implementations. In particular, BC is implemented on a realistic simulation of a differential drive robot, modelled on the SegwayRMP, performing a ball interception task. Our results show that human teacher critiquing does improve task performance, as measured by interception success and efficiency. A-OPI is implemented on a real Segway RMP robot performing a spatial positioning task. We show that A-OPI enables similar or superior performance when compared to the typical LfD approach to behavior correction that provides more teacher

B. Argall and B. Browning are with the Robotics Institute, and M. Velsoso the Computer Science Department, at Carnegie Mellon University, Pittsburgh, PA 15213, USA. <bargall,brettb,mveloso>@cs.cmu.edu

This research was sponsored in part by the Boeing Company under Grant No. CMU-BA-GTA-1. The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing official policies or endorsements, either expressed or implied of the Boeing Company. The authors Brenna Argall and Brett Browning were partially supported by Carnegie-Mellon University in Qatar and the Qatar Foundation.

demonstrations. Furthermore, by concentrating new data exclusively to the areas visited by the robot and needing improvement, A-OPI produces noticeably smaller datasets.

In the following section we motivate LfD policy improvement with a review of related work. Section III introduces our general algorithm. The BC implementation is presented in Section IV, along with validation details and results in Section V. The same treatment is given to the A-OPI implementation, in Sections VI and VII respectively. In Section VIII we conclude.

II. MOTIVATION AND RELATED WORK

The problem of learning a mapping between world observations and proper action selection lies at the heart of many robotics applications. Here we present an overview of the Learning from Demonstration approach to this problem, followed by a motivation for policy improvement based on learner experience within such a framework.

A. Learning from Demonstration

During an LfD teacher execution, world state observation and action selection is recorded. Formally, our world consists of states S and actions A, with the mapping between states by way of actions being defined by the probabilistic transition function $T(s'|s,a) : S \times A \times S \rightarrow [0,1]$. We assume that state is not fully observable. The learner instead has access to observed state Z, through a mapping $S \rightarrow Z$. A teacher demonstration $d_j \in D$ is represented as t_j pairs of observations and actions such that $d_j = \{(\mathbf{z}_j^i, \mathbf{a}_j^i)\} \in D, \mathbf{z}_j^i \in Z, \mathbf{a}_j^i \in A, i = 0 \cdots n_j$. Within the typical LfD paradigm, the set D of these demonstrations is then provided to the learner. No distinction is made within D between the individual executions however, and so for succinctness we notate $(\mathbf{z}_k, \mathbf{a}_k) \equiv \left(\mathbf{z}_j^i, \mathbf{a}_j^i\right)$. A policy $\pi : Z \rightarrow A$, that selects actions based on an observation of the current world state or *query point*, is then derived from D.

LfD has found success on a variety of robot platforms and applications. Key design decisions include the demonstration approach that produces the data, and then how a policy is derived from that data.

Approaches for executing and recording teacher demonstrations range from teleoperating a passive robot platform [12] to recording sensors worn by a human [7]. We teleoperate our robot while recording from its own sensors, as this minimizes correspondence issues between demonstrator and learner, and employ both human and automated demonstrators, as the algorithm is general to any sort of teacher.

The most popular approaches for deriving a policy from demonstration data are to (a) directly approximate the underlying function mapping observations to actions [5], (b) use the data to determine the world dynamics model $T(s'|s,a)$ [11] or (c) provide a planner with a learned model of action pre- and post-conditions [10]. Our work derives a policy using the first, mapping function approximation, approach.

B. Policy Improvement within Learning from Demonstration

An attractive feature for any learning system is the ability to update the policy based on learner executions. Though not a part of the classical LfD formulation, a variety of LfD systems do update the policy based on learner experience. For example, when a policy is derived under the world dynamics model approach, execution experience may update $T(s'|s,a)$ [1], or reward-determined state values [13].

When a policy is derived by approximating the underlying mapping function, which is the approach we explore, a common technique for improvement is to provide more demonstration data. Algorithms driven by learner requests for more data [8], [9], and teacher initiation of further demonstration [7], have been shown to improve policy performance.

In this work, we consider two alternative approaches to LfD policy improvement. The first (BC) attaches a valuation, based on teacher feedback, to the demonstration data points, and considers this valuation when deriving the policy. The second (A-OPI) provides more data to the policy, but derives this data from *learner* executions modified by advice from a human teacher (Fig. 1).

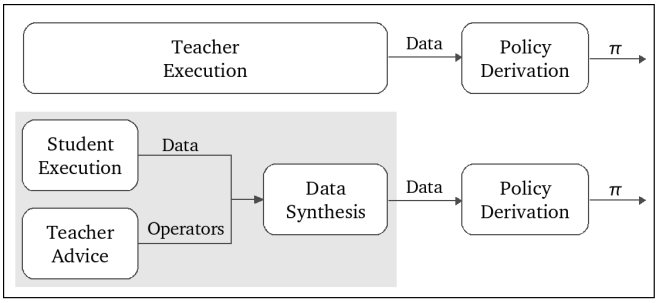

Fig. 1. Generating demonstration data. Typical approaches provide demonstration data from teacher executions (top). Our A-OPI approach introduces a novel technique for generating data based upon learner executions and teacher advice (bottom, shaded box).

III. GENERAL ALGORITHM

We now detail our general teacher feedback algorithm. The approach consists of two phases. During the *demonstration phase*, a set of teacher demonstrations is provided to the learner. From this the learner generalizes an initial policy. During the *advising phase*, the learner executes this initial policy. Advice on the learner execution is offered by a human advisor, and is used by the student to update its policy. Psuedo-code for the algorithm is provided in Figure 2.

The first phase consists of teacher demonstration, during which state-action pairs are recorded. From the set of these exeuctions D an initial policy is derived (line 1). The second phase consists of learner practice. For each practice run, an observed goal state $z^{goal} \in Z$ is provided (line 2). To begin the robot performs the task (*Execute*, lines 4-9).

During the *Execute* portion of a practice run, the learner first executes the task, producing trace tr. The learner executes until achieving goal state z^{goal}. At each timestep the learner selects action $\mathbf{a}^t$ according to $\pi(\mathbf{z}^t)$ (line 6).

```
0   Given  D, Z
1     initialize  π ← policyDerivation(D)
2     Given  z^goal ∈ Z
3       initialize  tr ← {}
4       Execute
5         while z^t != z^goal
6           select    a^t ← π(z^t)
7           execute   a^t
8           record    tr  ← { z^t, a^t }
9         end
10      Update
11        update    D ← adviseDataset(tr, D)
12        rederive  π ← policyDerivation(D)
```

Fig. 2. Psuedo-code for the general teacher feedback algorithm.

This action is recorded in the execution trace tr, along with observation $\mathbf{z}^t$ (line 8).

The advisor then provides feedback on this performance (line 11). The exact form taken by this advice, and how it is used by the learner to update its policy, is determined by each algorithm distinctly and will be discussed further in Sections IV and VI for BC and A-OPI respectively. The learner updates its policy π in response to the data feedback of the advisor (*Update*, lines 12-15).

Lastly, in our implementation the policy is derived using regression techniques. This is due jointly to the continuous action space of our domains and that we employ the mapping function approximation approach to policy derivation. A wealth of regression approaches exist, and we emphasize that many are compatible with this algorithm. Regression tuning is external to this algorithm and tied to policy derivation, occurring for initial derivation (line 1) and possibly for re-derivations (line 12).

IV. ALGORITHM: BINARY CRITIQUING

We now present the details of the Binary Critiquing implementation of our algorithm.

A. Algorithm Details

Within BC, how teacher feedback influences the policy update depends upon the regression approach used. In particular, teacher feedback is used to credit the underlying demonstration data. We use 1-Nearest Neighbors (1-NN) regression because it is straightforward to determine which datapoints were involved in a particular prediction, a key requirement for this algorithm.

Updating the control policy depends on an internal data representation, that is constructed by associating a scaling factor m_i with each training set observation-action pair $(\mathbf{z}_i, \mathbf{a}_i) \in D$. This factor scales the distance computation during the 1-NN prediction. Formally, given a current observation $\mathbf{z}^t$, the scaled distance to each observation point $\mathbf{z}_i \in D$ is computed, according to some metric $\|\mathbf{z}^t - \mathbf{z}_i\|$. The minimun is determined, and the action in D associated with this minimum

$$\mathbf{a}^t \;=\; \mathbf{a}_{\arg\min_i \|\mathbf{z}^t - \mathbf{z}_i\| m_i}, \;\; (\mathbf{z}_i, \mathbf{a}_i) \in D \qquad (1)$$

is executed. In our implementation the distance computation is Euclidean and observation dimensions are scaled inversely with range. The values m_i are initially set to 1.

Psuedo-code for dataset updating based on human advice within the BC algorithm is presented in Figure 3. This code chunk ties to line 11 of Figure 2, and will be discussed within the following section.

```
0  adviseDataset(tr, D)
1    Advise
2      { d̂ } ← teacherFeedback(tr)
3    Modify
4      foreach (z^k, a^k) ∈ d̂
5        choose   (z_i, a_i, m_i) ∈ D that recommended a^k
6        update   m_i ←  m_i + κ||z^t − z_i||^−1
7      end
8  return D
```

Fig. 3. Psuedo-code for dataset advising within the BC algorithm.

B. Teacher Feedback: A Binary Valuation

Within the BC algorithm, teacher feedback consists of a binary critique. For each learner execution, the teacher selects chunks $\hat{d}$ of the trajectory to flag as poorly performing (Fig. 3, line 2). Chunk sizes are determined dynamically by the teacher, and may range from a single point to all points in the trajectory (typical sizes in this work are 20-30 points).

For each point k flagged by the teacher, the algorithm determines the point $(\mathbf{z}_i, \mathbf{a}_i, m_i) \in D$ that recommended the action $\mathbf{a}^k$ executed by the learner (line 5). The value m_i of this point is then increased according to line 6, where $\kappa > 0$ is some constant. The amount by which m_i is increased is scaled inversely with distance, so that points will not be unjustly penalized if recommending for remote areas of the observation space. To update m_i in this manner means that datapoints whose recommendations gave poor results (according to the critique of the teacher) will be seen as farther away during the nearest neighbor distance calculation.

We conclude this section with a comparison of our teacher feedback to traditional *Reinforcement Learning (RL)* reward. In BC, poor performance flags provided by the teacher result in a devaluation of state-action points within the demonstration dataset; effectively, a decrease in accumulated state reward, or *value*. This valuation is different from traditional RL reward however, both in how it is provided and how it is used. First, BC feedback is provided by a human teacher, whose evaluations may provide richer feedback than the simple sparse reward functions that typically provide RL rewards. Second, accumulated reward is not considered in a *predictive* sense, as it is in RL where the action that leads to the highest valued state is selected. Rather, state value is used to modify the support of the demonstration dataset, such that low-value data points are no longer considered to support the state-space in which they reside. Note that a similar approach to value use is taken by Bentivegna [6], where the Q-value of a combined state-action-*query* point is automatically decreased if the point resulted in task failure.

V. EMPIRICAL VALIDATION: BINARY CRITIQUING

In this section we present results from applying the BC algorithm to a simulated motion interception task.

A. Simulated Motion Interception

Empirical validation of the BC algorithm is performed within a simulated ball interception domain. We first present the task and domain, and then our evaluation measures.

1) Task and Domain: A differential drive robot is tasked with intercepting a moving ball. Task execution ends when the robot either intercepts the ball or the ball travels out of bounds. Care was taken to keep the simulated domain realistic to a real world domain; further domain and implementation details are described in [2].

Policy executions begin from an initial world configuration of relative ball position and velocity. During execution the robot directly observes its own state and the ball position. We define (d^t, ϕ^t) as the distance and angle to the ball in the robot-centric frame, and (d^t_R, ϕ^t_R) as the distance traveled by and heading of the robot within the global world frame. The observations computed by the robot are 6-dimensional: $[d^t, \phi^t, (d^t - d^{t-1}), (\phi^t - \phi^{t-1}), (d^t_R - d^{t-1}_R), (\phi^t_R - \phi^{t-1}_R)]$. The actions predicted by the robot are 2-dimensional: change in rotational speed and change in translational speed.

Teacher demonstrations are performed via teleoperation of the robot by a hand-written suboptimal controller able to select changes in rotational and translational speed. This teacher was chosen for two reasons: the first being the ease with which a large number of demonstrations could be provided (here 100), and the second to highlight the ability of our algorithm to improve upon teacher suboptimality. For teacher demonstrations, initial world configurations are uniformly sampled.

2) Evaluation: To measure the performance of our algorithm, trajectory executions are evaluated for success and efficiency. A successful interception is defined by (a) the relative distance to the ball falling below some threshold and (b) the ball and robot both remaining within bounds. Execution efficiency is measured by trajectory length.

Performance evaluations occur on an independent test set containing n_t randomly sampled initial world conditions ($n_t = 100$). Execution on this test set is carried out after every n_p practice executions, or *critiquing rounds* ($n_p = 20$, 120 critiquing rounds in total). We define a critiquing round as execution from a single randomly sampled initial world condition (not found within the test set). Note that during the test evaluations, the learner executes using its most recent policy π, but *no* teacher critiquing or policy updating occurs. For comparative purposes, the performance of the demonstration teacher on this test set is evaluated as well.

B. Empirical Results

In this section we show learner performance to improve with critiquing. This performance improvement is shown through an increase in interception successes, as well as the more efficient execution of successful trajectories. On both of these measures, learner performance not only improves, but comes to exceed the performance of its teacher. An example cycle of learner execution and teacher critique, along with the subsequent improvement in learner execution, is demonstrated in Figure 4.

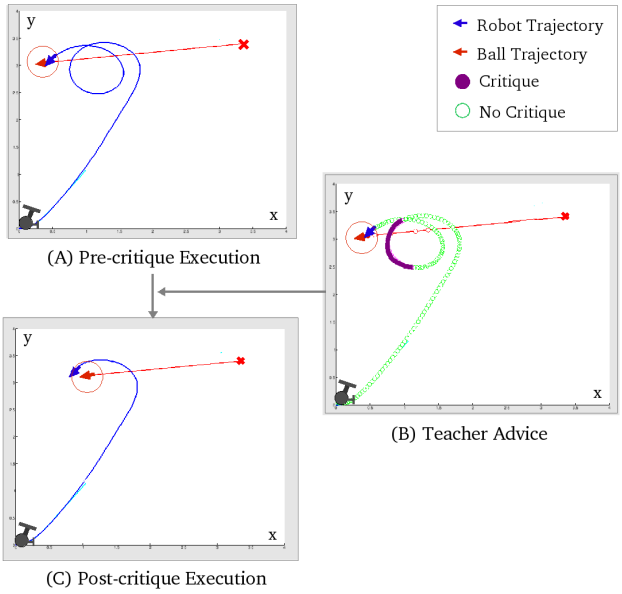

Fig. 4. Example learner execution made more efficient by critiquing. (A) The robot initially intercepts the ball, but the loop in its trajectory is inefficient. (B) The teacher critiques this trajectory, flagging the loop as poor. (C) The robot repeats the execution successfully without a loop. Arrow heads indicate direction of travel, and the red circle the distance threshold for successful interception.

1) More Successful Executions: Learner performance was found to improve with teacher critiquing. This improvement was seen on advised executions, as well as under validation by an independent test set.

Policy development was marked by rounds of critiquing practice. Testing with the independent test set was performed intermittently between rounds, to mark learner progress. Figure 5 shows learner improvement, where each data point represents the average result of executing from all test set initial conditions. Learner percent success, using the initial and final policies, are shown in Table I.

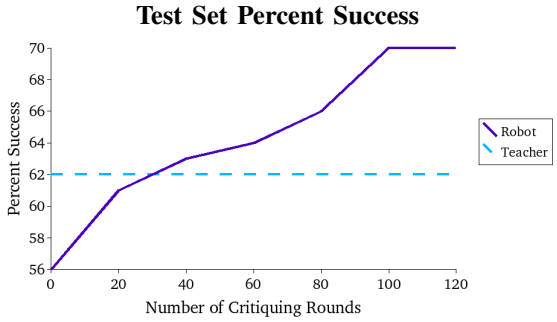

Fig. 5. Improvement in learner performance on the test set with critiquing rounds (solid line). (Hand-written teacher policy test set performance provided for comparison, dashed line).

For comparison, the teacher also executed from the test set initial world conditions (Fig. 5, Tbl. I). That the learner was able to perform *better* than the demonstrator underlines the benefits of critiquing within this domain. The hand-coded demonstration controller is not optimal for the domain. By critiquing the robot's executions, BC is able to correct for some demonstration error and improve the robot's performance beyond the capabilities of the demonstration teacher, and all in a simple and straightforward manner.

2) More Efficient Executions: Critiquing also improved learner execution efficiency. That is, the robot learned to intercept the ball faster, indicated by a reduction in trajectory length. Efficiency results on the independent test set, from the learner executing with its initial and final policies, as well as the teacher executions, are presented in Table I. Note that to decouple this measure from success, we compare only those runs in which both learner and teacher are successful (since the out of bounds success measure otherwise taints the comparison, as a successful interception is necessarily faster than an unsuccessful one).

	Learner Initial π	Learner Final π	Teacher
Success	56%	**70%**	62%
Length	2.73	**1.96**	2.87

TABLE I. Execution Percent Success

C. Discussion

The evaluation criteria of the feedback teacher was a combination of ball interception success and human intuition. These criteria depended heavily upon the teacher determining when the execution began to 'go wrong', and passing judgement on whether the robot was doing something 'smart'. For example, taking a very convoluted path to the ball would be considered 'not smart', even if the interception was successful (Fig. 4). To formally define a metric for credit assignment which determines 'wrongness' and 'smartness', however, could be quite difficult, thus underlining the worth in having the critiquing teacher be *human*.

One weakness of this algorithm is that points in D might be unduly penalized by critiquing, since *where* query points are in relation to training data points is not considered when updating m_i. Consider two query points located at identical distances, but orthogonal directions with respect, to a given training point. The training point's recommended action might be appropriate for one query point but not the other. Its execution by each would incur different successes, and therefore also conflicting critiques. The incorporation of query point orientation into the update of m_i is thus a potential improvement for this algorithm.

Providing a binary critique, like providing RL reward, gives the learner an indication of where poor action selection occurred. It does not, however, provide any sort of indication about what should have occured instead. We posit that more informative, *corrective*, feedback would prove useful to the learner, and upon this ground our development of teacher feedback within the A-OPI implementation.

VI. ALGORITHM: ADVICE-OPERATOR POLICY IMPROVEMENT (A-OPI)

We next present the details of the Advice-Operator Policy Improvement implementation of our algorithm. We begin with a brief discussion on correcting behavior within LfD. Typical approaches have the teacher *demonstrate* the correct behavior, and then rederive the policy with this new data. Within A-OPI we take the novel approach of *synthesizing* new demonstration data based on student executions and teacher advice. The following considerations motivate our interest in alternatives to teacher demonstration:

- *No need to recreate state.* This is especially useful if the world states where demonstration is needed are dangerous (e.g. lead to a collision), or difficult to access (e.g. in the middle of a motion trajectory).
- *When unable to demonstrate.* Further demonstration may actually be impossible (e.g. rover teleoperation over a 40 minute Earth-Mars communications lag).
- *Not limited by demonstrator.* A demonstrated training set is inherently limited by the demonstrator's performance, which may be suboptimal.

A. Algorithm Details

Within A-OPI, the policy is updated by considering new data, which has been synthesized from teacher advice and learner executions. Unlike BC, the incorporation of teacher feedback does *not* depend on the particulars of the regression technique, and any may be used. Our implementation employs a form of Locally Weighted Learning [4]. Given current observation $\mathbf{z}^t$, action $\mathbf{a}^t$ is predicted through an averaging of data points in D, weighted by their kernelized distance to $\mathbf{z}^t$. Thus,

$$\mathbf{a}^t = \sum_{(\mathbf{z}_i,\mathbf{a}_i)\in D} w_i^t \cdot \mathbf{a}_i, \quad w_i^t = e^{\|\mathbf{z}_i - \mathbf{z}^t\|} \tag{2}$$

where the weights w_i^t have been normalized over i. In our implementation the distance computation is Euclidean, observation dimensions are scaled by constant parameters (tuned through cross-validation) and the kernel is Gaussian.

Psuedo-code for dataset updating based on human advice within the BC algorithm is presented in Figure 6. This code chunk also ties to line 11 of Figure 2, and will be discussed within the following section.

B. Teacher Feedback: A Continuous Correction

The purpose of advice within A-OPI is to correct the robot's policy. Though this policy is unknown to the human advisor, it is represented by observation-action mapping pairs. To correct the policy, our approach therefore offers corrective information about observation-action pairings from a learner execution. A key insight to the A-OPI approach is that pairing a modified observation (or action) with the *executed* action (or observation) now represents a corrected mapping. Assuming accurate policy derivation techniques, adding this data point to the demonstration set and re-deriving the policy will thus also correct the policy.

```
0   adviseDataset(tr, D)
1     Advise
2       { op, d̂ } ← teacherFeedback(tr)
3     Modify
4       foreach (z^k, a^k) ∈ d̂
5         if op is observation-modifying
6           record D ← { op(z^k), a^k }
7         else op is action-modifying
8           record D ← { z^k, op(a^k) }
9         end
10      end
11  return D
```

Fig. 6. Psuedo-code for dataset advising within the A-OPI algorithm.

To have a human provide this corrective information, however, represents a significant challenge. Specifically, our work focuses on robot motion control within continuous state/action spaces. Correcting the state-action mapping of a policy involves indicating the correct action or state. For continuous spaces, this requires providing a continuous-valued correction. Expecting the human teacher, however, to know an appropriate continuous *value* that corrects these data points is neither reasonable nor efficient.

To circumvent this, we have developed *advice-operators* as a language through which the human teacher provides advice to the robot student. We concretely define an advice-operator as a mathematical computation performed on an observation input or action output. Key characteristics of advice-operators are that they:

- Perform mathematical computations on data points.
- Are defined commonly between the student and advisor.
- May be applied to observations or actions.

We have developed a set of 9 advice-operators for motion control (see [3] for full details).

Like in BC, the teacher first indicates a chunk $\hat{d}$ of the learner execution trajectory requiring improvement. The teacher additionally selects an advice-operator *op*, from a finite list, to correct the execution within this chunk (Fig. 6, line 2). For each point k selected by the teacher, the algorithm modifies either its observation (line 6) or action (line 8), depending on the indicated advice-operator type. The modified data points are then added to the set D.

To illustrate, consider as an example that the operator "Translational acceleration" is indicated along with a chunk of 10 execution data points. This operator functions by augmenting actions by linearly increasing percentages of the executed speed; for example updating the translational speed a^0 of point 0 to $a^0 \leftarrow 1.1 \cdot a^0$, point 1 to $a^1 \leftarrow 1.2 \cdot a^1$, and so forth through to the final point 9 to $a^9 \leftarrow 2.0 \cdot a^9$.

We again conclude with a comparison of this teacher feedback to traditional RL reward. In A-OPI, teacher advice provides a *correction* on the executed state-action mapping. By contrast, RL reward provides only an indication of the desirability of visiting a particular state; to determine the correction (i.e. the more desirable state) alternate states must be visited. This can be unfocused and intractable to optimize when working on real robot systems with an infinite number of world state-action combinations.

VII. EMPIRICAL VALIDATION: A-OPI

In this section we present the results of applying the A-OPI algorithm to a motion task using a Segway RMP robot.

A. Segway RMP Spatial Positioning

Empirical validation of the A-OPI algorithm is performed through spatial positioning with a Segway RMP robot (Fig. 7). We first present the task and domain, and then our evaluation measures.

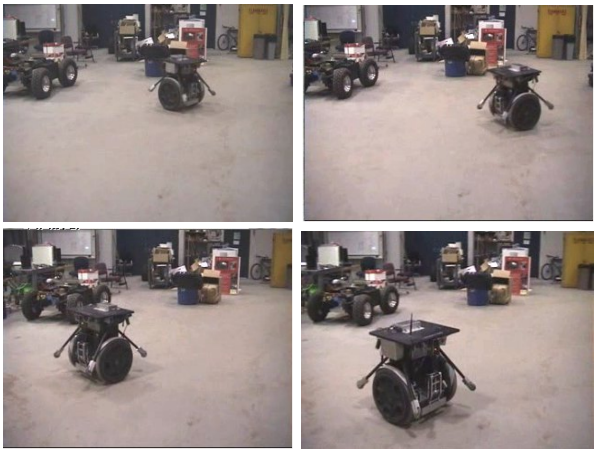

Fig. 7. Segway RMP robot performing the spatial positioning task.

1) Task and Domain: The Segway RMP is a dynamically balancing differential drive robot produced by Segway LLC. The platform accepts wheel speed commands, but does not allow access to its balancing control mechanisms. The inverted pendulum dynamics of the robot present an additional element of uncertainty for low level motion control. Furthermore, for this task smoothly coupled rotational and translational speeds are preferred, in contrast to turning on spot to θ_g after attaining (x_g, y_g). To mathematically define such trajectories for this specific robot platform is thus non-trivial, encouraging the use of alternate control approaches such as A-OPI. That the task is straightforward for a human to evaluate and correct further supports A-OPI as a candidate approach. While the task was chosen for its suitability to validate A-OPI, to our knowledge this work also constitutes the first implementation of such a motion task on a real Segway RMP platform.

The spatial positioning task consists of attaining a 2D planar target position (x_g, y_g), with a target heading θ_g. The observations and actions for this task are 3- and 2-dimensional, respectively. Let the current robot position and heading within the world be represented as (x_r, y_r, θ_r), and the vector pointing from the robot position to the goal position be $\langle x_v, y_v \rangle = \langle x_g - x_r, y_g - y_r \rangle$. An observation consists of: squared Euclidean distance to the goal $(x_v^2 + y_v^2)$, the angle between the vector $\langle x_v, y_v \rangle$ and robot heading θ_r, and the difference between the current and target robot headings $(\theta_g - \theta_r)$. An action consists of: translational and

rotational speeds. The robot samples these values from wheel encoders at 30 Hz.

2) Policy Development: The set D is seeded with demonstrations recorded as the teacher teleoperates the robot learner (here 9 demonstrations, 900 data points). The initial policy derived from this dataset we refer to as the *Baseline Policy*.

Policy improvement proceeds as follows. A goal is selected (without replacement) from a practice set consisting of (x, y, θ) goals drawn uniformly within the bounds of the demonstration dataset $([-0.33, 4.5]m, [-4.0, 0.17]m, [-3.1, 1.1]rad)$. The robot executes its current policy to attain this goal. The advisor observes this execution. If the execution is considered poor, the advisor offers policy improvement information. The policy is re-derived. Drawing a new goal then initiates another practice cycle.

Three policies are developed using distinct techniques, differing in *what* is offered as policy improvement information. The first provides advice exclusively, in the form of advice-operators (*A-OPI Advised Policy*). The second involves an initial phase of exclusively more teleoperation, followed by a phase of exclusively offering advice (*A-OPI Hybrid Policy*). The third provides further teleoperation teacher demonstrations exclusively (*Teleoperation Policy*). We refer to these collectively as the *improvement policies*.

3) Evaluation: Policy performance is evaluated on a test set, consisting of 25 (x, y, θ) goals, again drawn from a uniform distribution within the bounds of the demonstration dataset. The test set is independent, and no executions associated with it receive policy improvement information.

Policies are evaluated for accuracy and success. *Accuracy* is defined as Euclidean distance between the final robot and goal positions $e_{x,y} = \|(x_g - x_r, y_g - y_r)\|$, and the final robot and goal headings $e_\theta = |\theta_g - \theta_r|$. We define *success* generously as $e_{x,y} < 1.0\ m\ \wedge\ e_\theta < \frac{\pi}{2}\ rad$.

B. Empirical Results

Policy performance improved with A-OPI advising, in both execution success and accuracy. When compared to the approach of providing more teleoperation data, final improvement amounts were found to be similar or superior. Furthermore, this performance was achieved with a similar number of practice executions, but smaller final dataset D.

For each policy improvement approach, the policy improvement phase was halted once performance on the test set no longer improved. The final A-OPI Advised, A-OPI Hybrid and Teleoperation Policies contained data from 69, 68 and 60 executions, respectively (with the first 9 demonstrations for each attributable to seeding with the Baseline Policy).

1) Increase in Successful Executions: Figure 8 presents the percent execution success of each policy on the independent test set. When compared to the Baseline Policy, all policy improvement approaches display increased success. Both the advised A-OPI policies additionally achieve higher success than the Teleoperation Policy.

Test Set Percent Success

A-OPI Advised	A-OPI Hybrid	Teleop	Baseline
88%	92%	80%	32%

Fig. 8. Percent successfully attained test set goals.

2) Improved Accuracy: Figure 9 plots the average position and heading error on the test set goals, for each policy. For positional error, all improvement policies display similar performance, which is a dramatic improvement over the Baseline Policy. For heading, A-OPI Advised reduces more error than A-OPI Hybrid, with both showing marked improvements over the Baseline Policy. By contrast, the Teleoperation Policy displays *no* overall improvement in heading error.

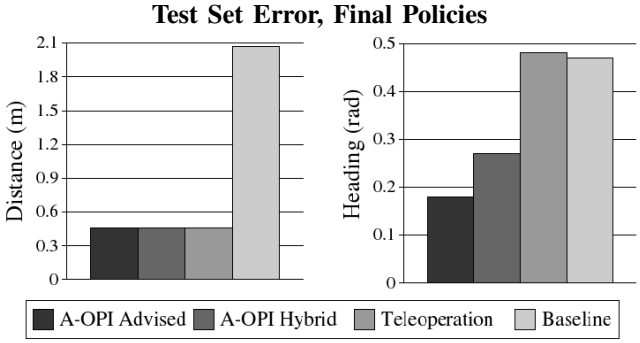

Fig. 9. Average test set error on target position (left) and heading (right), with the *final* policies.

That heading error proved in general more difficult to improve than positional error is consistent with our prior experience with this robot platform being highly sensitivity to rotational dead reckoning error accumulation.

A sampling of the intermediate policies' performance is provided in Figure 10. For a more complete discussion of these results, we refer the reader to [3]. Do note, however, that these results are plotted against the number of *executions* contributing to the set D, and not the number of *data points*.

3) More Focused Improvement: How many data points are added with each execution varies greatly depending upon whether the execution is advised or teleoperated (Fig. 11). This is because, in contrast to teleoperation, only subsets of an advised execution are added to D; in particular, only those execution points which actually receive advice. States visited during good performance portions of the student execution are *not* redundantly added to the dataset. In this manner, the final policy performances shown in Figure 9 are achieved with much *smaller* datasets for both A-OPI policies, in comparison to the Teleoperation Policy.

C. Discussion

The empirical results confirm that a human teacher was able to effectively provide advice within *continuous* action spaces. This occurred without the teacher providing the continuous-values for these corrections, or requiring value-

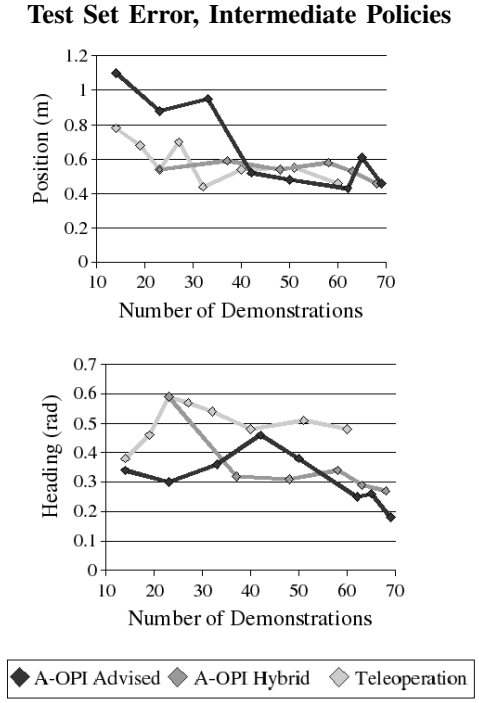

Fig. 10. Average test set error on target position (top) and heading (bottom), with *intermediate* policies (shown against the number of advised and/or teleoperated demonstrations).

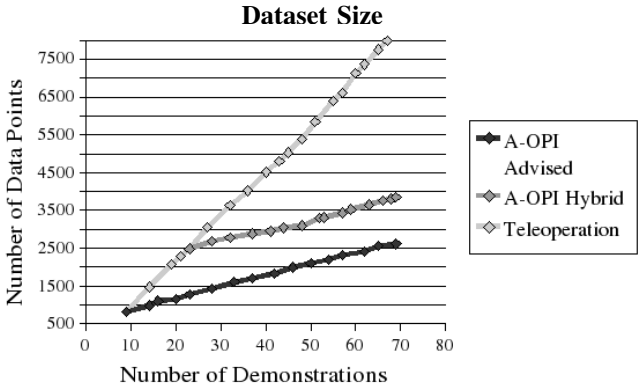

Fig. 11. Dataset size growth with demonstration number.

based execution details (e.g. speed) to select operators. The robot, and not the human, applied the operator.

The teacher was able to provide advice quickly, because the algorithm is robust to fuzzy selections of execution points. Since regression treats each data point independently, a point added to the dataset does not depend upon whether nearby points from the execution were also added. Furthermore, operators are "smart" enough to check for particular data point qualities when necessary. For example, a point which already had zero rotational speed would be skipped by the "No turning" operator.

We finish with comments on the hybrid policy. The hybrid policy was seeded with an intermediate Teleoperation Policy and then advised, in an attempt to exploit the strengths of each approach. One direction for future work could interleave providing advice and more teleoperation. Alternately, a sufficiently populated demonstration set could be provided at the start. This is fundamentally different from providing more teleoperation in response to *student* executions, however. It requires prior knowledge of all states the student will visit; generally impossible in continuous-state real worlds.

VIII. CONCLUSION

We have presented an algorithm for policy improvement based on human advice within a Learning from Demonstration framework. We discuss two implementations of this algorithm, that differ in the type of feedback provided. The first implementation, Binary Critiquing (BC), provides the learner with an indication of poor performance areas. We validate this implementation in a realitic robotic simulation and demonstrate BC to improve task performance. In both execution success and efficiency, learner performance not only improved but came to *exceed* the performance of its demonstration teacher. The second implementation, Advice-Operator Policy Improvement (A-OPI), is distinguished by providing *continuous* corrections on learner executions, as well as an alternative data source to teacher demonstrations. We validate this implementation on a real robot system. Policy modifications due to A-OPI were shown to improve policy performance on a Segway RMP robot, both in execution success and accuracy. Furthermore, performance was found to be similar or superior to the typical LfD approach of correcting behavior through more teacher demonstrations.

REFERENCES

[1] P. Abbeel and A. Y. Ng. Exploration and apprenticeship learning in reinforcement learning. In *Proceedings of ICML '05*, 2005.
[2] B. Argall, B. Browning, and M. Veloso. Learning from demonstration with the critique of a human teacher. In *Proceedings of HRI '07*, 2007.
[3] B. Argall, B. Browning, and M. Veloso. Learning robot motion control with demonstration and advice-operators. In *Proceedings of IROS '08*, 2008.
[4] C. G. Atkeson, A. W. Moore, and S. Schaal. Locally weighted learning for control. *Artificial Intelligence Review*, 11:75–113, 1997.
[5] D. C. Bentivegna. *Learning from Observation Using Primitives*. PhD thesis, College of Computing, Georgia Institute of Technology, 2004.
[6] D. C. Bentivegna. *Learning from Observation using Primitives*. PhD thesis, Georgia Institute of Technology, 2004.
[7] S. Calinon and A. Billard. Incremental learning of gestures by imitation in a humanoid robot. In *Proceedings of HRI '07*, 2007.
[8] S. Chernova and M. Veloso. Confidence-based learning from demonstration using Gaussian Mixture Models. In *Proceedings of AAMAS '07*, 2007.
[9] D. H. Grollman and O. C. Jenkins. Dogged learning for robots. In *Proceedings of ICRA '07*, 2007.
[10] M. N. Nicolescu and M. J. Mataric. Methods for robot task learning: Demonstrations, generalization and practice. In *Proceedings of AAMAS '03*, 2003.
[11] E. Oliveira and L. Nunes. *Learning by exchanging Advice*. Springer, 2004.
[12] P. K. Pook and D. H. Ballard. Recognizing teleoperated manipulations. In *Proceedings of ICRA '93*, 1993.
[13] M. Stolle and C. G. Atkeson. Knowledge transfer using local features. In *Proceedings of ADPRL '07*, 2007.