Learning by Demonstration with Critique from a Human Teacher *

Brenna Argall
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, PA, USA
bargall@cs.cmu.edu

Brett Browning
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, PA, USA
brettb@cs.cmu.edu

Manuela Veloso
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, PA, USA
mmv@cs.cmu.edu

ABSTRACT

Learning by demonstration can be a powerful and natural tool for developing robot control policies. That is, instead of tedious hand-coding, a robot may learn a control policy by interacting with a teacher. In this work we present an algorithm for learning by demonstration in which the teacher operates in two phases. The teacher first demonstrates the task to the learner. The teacher next critiques learner performance of the task. This critique is used by the learner to update its control policy. In our implementation we utilize a 1-Nearest Neighbor technique which incorporates both training dataset and teacher critique. Since the teacher critiques performance only, they do not need to guess at an effective critique for the underlying algorithm. We argue that this method is particularly well-suited to human teachers, who are generally better at assigning credit to performances than to algorithms. We have applied this algorithm to the simulated task of a robot intercepting a ball. Our results demonstrate improved performance with teacher critiquing, where performance is measured by both execution success and efficiency.

Categories and Subject Descriptors

I.2.9 [**Artificial Intelligence**]: Robotics

General Terms

Algorithms, Experimentation

1. INTRODUCTION

Learning by demonstration can be an effective tool for the development of policies to control robot behavior. We argue that teaching by demonstration is a natural approach for a human teacher, as humans already employ demonstration to teach other humans. As such, it enriches our knowledge of human-robot interactions, while making robotics more accessible to the general public. In comparison to the alternative of hand-coding, demonstration opens robot policy development to non-experts in the field. Beyond providing the demonstration, a human teacher may additionally help a robot learner by offering a critique on performance. We argue that this role is also well-suited for human teachers, who are generally good at evaluating task performance.

When learning by demonstration, a robot learns a control policy from the demonstrated actions of a teacher. Most work in this area places the majority of the learning burden on the robot, with an exception being the work of Nicolescu and Mataric [11]. One way a teacher can help shoulder some of this burden is by commenting on learner performance. In our presented algorithm, the robot first derives a control policy from the demonstrations of a teacher. The teacher then indirectly modifies the learner's control policy through a critiquing process. In particular, the learner, and not the teacher, applies the teacher's critique to the policy update. We argue that this indirect policy modification technique is well-suited to human teachers. That is, humans are generally good at assigning performance credit, but have less intuition about assigning credit to underlying algorithms. Additionally, a robot learner and human teacher will undoubtedly differ in physics and logic. Under this formulation, a robot mechanism does not need to be performable, or understood, by the human teacher to receive credit.

We contribute an algorithm for learning a robot's control policy, in which the teacher provides both task demonstrations and performance critiques. We argue that the critiquing task is natural and appropriate for a human teacher, and simpler than hand coding a control policy. To validate our approach we implemented 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. We measure task performance by interception success and efficiency.

In the next section, we present background information and review related work. Section 3 formalizes this learning problem, while Section 4 describes our algorithm. In Section 5 the problem domain is presented, and in Section 6 experiemental results. In the final section we conclude.

2. RELATED WORK

Applications of machine learning to robot behaviors occur

*This work was supported by United States Department of the Interior under Grant No. NBCH-1040007. The content of the information in this publication does not necessarily reflect the position or policy of the Defense Advanced Research Projects Agency (DARPA), US Department of Interior, US Government, and no official endorsement should be inferred.

HRI'07, March 10–12, 2007, Arlington, Virginia, USA.

at many levels, from high level reasoning [9] to low level motion control policies [4]. In this paper, we focus on learning low level skills by demonstration.

Teaching a robot by demonstration provides training data in the form of example executions by a teacher. A task is executed by a teacher, and the details of this execution, usually in the form of paired observations and actions, are passed on to the learner. The learner then generalizes from these demonstrations in order to effectively execute the task itself. Work which has applied demonstration learning to the development of low level robot control policies includes: [3] learned the reward function used in control policy computations, [6] learned a heirarchy of neural networks for motion control, and [10] represented motor behaviors sparsely with via points. Local learning was used for motion control policy development by Atekson et al. [2]. The work presented in this paper also uses local learning techniques, and like [5, 8] combines this with teaching by demonstration.

In general, the example executions of a teacher will not apply directly to a learner. Many techniques rely upon an unknown mapping from teacher observations and actions to robot observations and actions. This correspondence issue has been addressed by [1, 12, 13], with [7] circumventing part of this problem by having the learner follow the teacher and record its own observations. Furthermore, the demonstration data provided by the teacher may not be well-suited for the generalization techniques employed by the learner. Other difficulties arise when good teacher data is unavailable for certain parts of the observation space, though we do not address this issue in this work. Any of these reasons may lead to poor learner performance.

One way to help the learner is to offer a critique on its performance. Helping the learner in this manner is a topic which has been explored at a high level, with the goal of social interactions for their robot, by Nicolescu and Mataric [11]. The algorithm presented in this paper similarly makes uses of the advice of a teacher. However, our approach incorporates the critique of a teacher at a low level, into the motion control policy computation. The use of a critique is also a key difference between our approach and traditional reinforcement learning frameworks. By having the teacher reward learner executions, we exploit the use of a human to overcome the challenge of credit assignment.

3. PROBLEM FORMALISM

To formally describe our teaching approach, we first make use of the Markov Decision Process (MDP) framework to define the robot control problem, and then we will introduce the teacher. Thus, we have a robot repeatedly taking actions in the world according to some control policy and observing their outcome, with the goal of maximizing the reward it collects. More formally, let us describe the problem as the tuple $< S, A, P, R >$, where S defines the set of states describing the robot and its world, A is the set of actions the robot can perform, and $P : S \times A \times S \rightarrow \Re$ is a probability function describing the state-action transitions. Thus, at each time step the state of the world is given by $s^t \in S$ and the robot selects some action $a^t \in A$. The probability of being in a given state using a 1-step Markov independence assumption is then defined by:

$$P\left(s^{t+1}|s^t, a^t, s^{t-1}, a^{t-1}, \cdots, a^0\right) = P\left(s^{t+1}|s^t, a^t\right) \quad (1)$$

In our framework, we assume that the robot cannot directly observe the state of the world. Rather, the robot observes a mapping of the world state defined as $H : S \rightarrow O$, a mapping from the world states onto O, the set of observations. We assume that the mapping is sufficiently rich for the robot to perform the task even with this mapping. The robot chooses actions based on its observation of the world state with its control policy $\pi : O \rightarrow A$ (see Figure 1A). The reward function $R : S \times A \rightarrow \Re$ defines the payoff achieved by the robot given the state of the world and the action taken. Learning for the robot consists of changing its control policy π so as to increase the reward it collects.

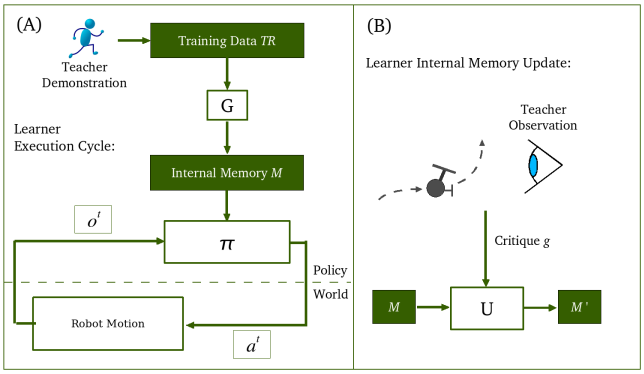

Figure 1: (A) A single time step of learner control policy execution. (B) Update of internal memory M using teacher critique.

In this paper, we focus on the problem of teaching rather than reinforcement learning. Thus, we introduce a teacher that is able to interact with the robot in two modes: by demonstrating an example, and by critiquing the robot's performance on a given example. We do not consider the mapping problem, and so demonstration consists of giving the robot a set of performed trajectories. Formally, teacher executions produce trajectories, which are sequences of state-action pairs showing the trace of the teacher's performance when starting from some initial condition in the world. The teacher provides the robot with the set $TR = \{tr_1, \cdots, tr_K\}$ derived from these trajectories, where $tr_i = ((o_i^0, a_i^0), \cdots, (o_i^{k_i}, a_i^{k_i}))$ and each (o_i^t, a_i^t) pair $o_i^t \in O, a_i^t \in A$ describe the observation and corresponding action selected at time t. Note that $o_i^0 = H\left(s_i^0\right)$, where $s_i^0 \in S$ is the initial world state. We will refer to these trajectories TR as the *training set*.

The second mode of interaction with the teacher is *critiquing*, whereby the robot attempts the problem starting from some initial state of the world and the teacher critiques the resulting learner execution trace. The teacher provides a signal $g \in \{0, 1\}$ that labels each time step as either good (1), or bad (0). Thus, the robot can form a critiqued trajectory for the test run as $tc = \left((o^0, a^0, g^0), \cdots, (o^L, a^L, g^L)\right)$. In our model the robot stores an internal memory M, from which its policy π is derived. M is seeded with the training set TR, such that $M = G(TR)$. It is updated according to $M' = U(M, g)$, and thus incorporates into π the critique stored in tc (Fig. 1B).

Finally, we formulate the teaching problem as: "To devise an algorithm to modify the robot's control policy π based on the demonstrated training set TR provided by the teacher and the memory M of critiqued robot performance, so as to

maximize its reward." The goal of our system is therefore to improve the learner policy, such that the summed reward from R is maximized.

Note that this approach makes the key assumption that the teacher has some unknown control policy $\Pi_{teach} : S \rightarrow A$ that can be represented in the robot's observation space as $\pi_{teach} : O \rightarrow A$ and still perform well (although not necessarily optimally).

4. ALGORITHM

In this section we present our algorithm for learning by demonstration with critiquing.

4.1 Overview

Learning for our system occurs in two phases. During the first *training phase*, example trajectories of observation-action pairs are built during teacher task execution. The set of these trajectories are provided by the teacher as the training dataset TR. This dataset will be used by the learner to form its internal representation M.

```
01  Given TR, I_critiq
02    M ← G(TR)
03  While !done_learning
04    sample s_i^0 ∈ I_critiq
05    While !teacher_satisfied
06      initialize world to s_i^0
07      While !task_done
08        execute a^t according to π = NN(o^t, M)
09      end
10      tc ← teacher critique of execution
11      For g^i ∈ tc
12        M ← U(M, g^i)
13      end for
14    end while
15  end while
```

Figure 2: Psuedo-code for the demonstration with critiquing algorithm.

In the second *critiquing phase*, the learner executes its control policy π and updates it based on the critique of the teacher (psuedo-code in Fig. 2). In particular, the learner begins by building an internal data representation M from TR. M is used to compute π using local learning techniques. To initialize the world, initial condition s_i^0 is sampled from the set $I_{critiq} \in S$ of initial states (explicitly described in Section 5.4). The learner executes π (lines 07-09). The teacher observes and critiques this execution, producing tc. Based upon teacher feedback tc, the learner updates M (lines 11-13) and subsequently π. The teacher may request the learner to execute this updated π, to monitor the effects of the critique. This cycle of execution-critique-update (lines 05-14) continues until the teacher is satisfied with learner performance. After this, a new initial condition s_i^0 is sampled from I_{critiq} to reinitialize the world, and learner execution begins again.

The entire algorithm of learning from demonstration with critiquing may therefore be described as follows:

1. *Train*
   (a) Teacher generates training dataset TR.
   (b) Learner derives M from TR, policy π from M.
2. *Critique*
   (a) Learner executes π from initial conditions.
   (b) Teacher observes and critiques learner performance, producing tc with labels g.
   (c) Learner updates memory M based on critique, thereby modifying π.
   (d) Repeat from (2a), to the satisfaction of the teacher.

With this algorithm, the learner uses the teacher's critique to modify its internal represenation of how the demonstration data is used. The critique is therefore applied in a manner meaningful to the learner, and appropriate to its control policy update. Having the learner apply the critique eliminates the effort and potential error in the teacher guessing effective ways to critique the actual policy derivation, or the actual policy update.

We exploit the use of a human to overcome the issue of credit assignment, by having the teacher reward an execution trajectory point. This is a key difference between our work and traditional reinforcement learning frameworks. The reward generated by the system is represented within the teacher's evaluation and critique of learner performance, and steps towards the maximization of this reward are taken when the learner incorporates the critique into its policy update. The learner itself, however, has no explicit concept of reward maximization. We thus try to optimize the expected average reward, as estimated by the teacher, and therefore do so in an indirect manner.

4.2 Learner Execution and Teacher Critique

The first part of the critiquing phase is marked by learner execution. The internal data representation M is constructed by associating a scaling factor m_i^j with each training set observation-action pair $\left(o_i^j, a_i^j\right) \in TR$. Thus,

$$M = \left\{ \begin{array}{c} \left(\left(o_1^1, a_1^1, m_1^1\right), \cdots, \left(o_1^{k_1}, a_1^{k_1}, m_1^{k_1}\right)\right), \\ \cdots, \\ \left(\left(o_K^1, a_K^1, m_K^1\right), \cdots, \left(o_K^{k_K}, a_K^{k_K}, m_K^{k_K}\right)\right) \end{array} \right\}.$$

The values m_i^j are initially set to 1.

Local learning techniques on M are used to derive the policy π. In particular, we used a 1-Nearest Neighbor (1-NN) approach because it has been successfully applied to other robot control problems [5], and is comparatively simple and straightforward to implement. When using 1-NN, the closest datapoint is found within the training set, according to some distance metric, and its associated action is selected for execution.

Formally, the learner constructs a current *query point* o^t from its observations. The scaled distance to each observation point $o_i^j \in M$ is computed, according to some metric D. The value m_i^j is the scaling factor. The minimun is determined,

$$a^t = arg\ min_{i,j} D(o^t, o_i^j, m_i^j), \quad (2)$$

and the action $a_i^j \in M$, associated with this minimum, is executed.

The second part of the critiquing phase is marked by teacher critique. For each learner execution, the teacher selects entire chunks of the trajectory to be flagged as good $g = 1$ or bad $g = 0$. Chunk sizes are determined dynamically by the teacher, and may range from a single point

to all points in the trajectory (typical chunk size in this work was on the order of 20-30 points). Though here it is binary, this critique may easily extend to incorporate continuous values. The learner then takes this information and updates its control policy by updating M. Specifically, a value $m_i^j \in M$ is increased if its associated action a_i^j was taken during a portion of the learner execution which has been flagged as poor. Formally, for each time step t of the learner execution trace, the algorithm determines the action $a_i^j \in M$ that was executed by the learner. We update m_i^j according to

$$m_i^j \leftarrow \left\{ \begin{array}{cc} m_i^j + \kappa \left[D(o^t, o_i^j, m_i^j) \right]^{-1} & if \quad g = 0 \\ m_i^j & if \quad g = 1 \end{array} \right\} \quad (3)$$

where $\kappa > 0$ is some constant. To monitor the effects of the critique, the teacher may then request that the learner repeat execution with the updated policy π.

To update m_i^j in this manner means that datapoints whose recommendations gave poor results (according to the critique of the teacher) will be seen as further away during the nearest neighbor distance calculation. This method of relating success to distance scaling is similar to that employed by Bentivegna in [5]. The amount by which m_i^j is increased is scaled inversely with distance, so that points will not be unjustly penalized if recommending for remote areas of the observation space.

5. EXPERIMENTS

In this section, we describe the experiment domain for our algorithm. To ground our explanation, and to validate our approach, we use a ball interception task. That is, a robot must learn to intercept a moving ball (Fig. 3). In particular, we look at interception using a differential drive robot. Since a differential drive robot may only drive forward or turn, and cannot go sideways, interception of a moving ball becomes a difficult task. It also depends heavily upon the dynamics of the world, and so writing an appropriate control policy can be challenging in the absence of accurate predictive world motion models.

These first results were gathered in simulation, but care was taken to keep the simulated domain realistic. We are currently in the process of applying this algorithm to a real robot platform (Fig. 4A).

5.1 Domain Description

The state of the world $S \in \Re^9$ is an 9-dimensional vector consisting of the differential-drive robot position, orientation, linear and rotational speed $s_R^t = \left(x_R^t, y_R^t, \theta^t, v^t, \omega^t\right)$ and the ball position and speed $s_B^t = \left(x_B^t, y_B^t, \dot{x}_B^t, \dot{y}_B^t\right)$.

The robot may directly make observations about its own state and ball position. Observations available to the robot were chosen to consist of $(\phi^t, \dot{\phi}^t, \dot{\theta}^t, d^t, \dot{d}^t, \dot{d}_R^t) \in O$ (Fig. 4B):

- ϕ^t Robot-relative ball angle
- $\dot{\phi}^t$ 1-time step difference $\phi^t - \phi^{t-1}$
- $\dot{\theta}^t$ 1-time step difference on the global robot angle $\theta^t - \theta^{t-1}$
- d^t Robot-relative ball distance $\|(x_B^t, y_B^t) - (x_R^t, y_R^t)\|$
- $\dot{d}^t$ 1-time step difference $d^t - d^{t-1}$
- $\dot{d}_R^t$ 1-time step difference on the global robot distance traveled $\|(x_R^t, y_R^t) - (x_R^{t-1}, y_R^{t-1})\|$

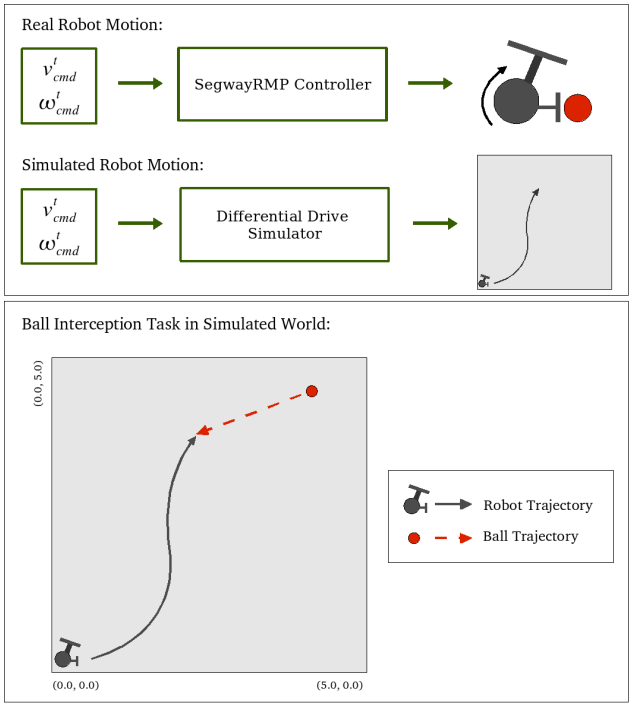

Figure 3: The motion of a real robot represented within our simulated world (top). A simulated ball interception task (bottom).

We assume the robot uses a differential-drive mechanism with a low-level controller that takes desired robot velocity input commands[1]. Thus, $a^t = \left(v_{cmd}^t, \omega_{cmd}^t\right)$. The actions encoded within the training set consist of $\Delta\omega_{cmd}^t$ and Δv_{cmd}^t, where

- $\Delta\omega_{cmd}^t$ 1-time step difference in commanded rotational robot speed $\omega_{cmd}^t - \omega_{cmd}^{t-1}$
- Δv_{cmd}^t 1-time step difference in commanded translational robot speed $v_{cmd}^t - v_{cmd}^{t-1}$

[1]While other arrangements are possible, this approach is common to many robot platforms

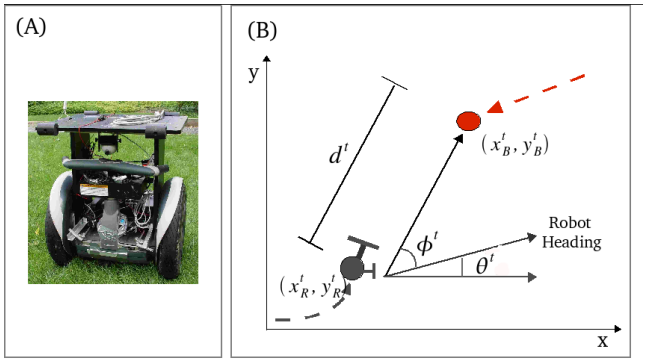

Figure 4: (A) A SegwayRMP robot. (B) State of the world observed by our simulated robot.

Note that within this formulation, observations and actions within the training set may be grouped according to which action command they influence. Specifically, $(\phi^t, \dot{\phi}^t, \dot{\theta}^t, \Delta\omega^t_{cmd})$ influence action command ω^t_{cmd}, and $(d^t, \dot{d}^t, \dot{d}^t_R, \Delta v^t_{cmd})$ influence v^t_{cmd}.

5.2 Simulated Dynamics

Here we describe the dynamics of our simulated domain. For the robot, motion is propagated by simple differential drive simulation (Fig. 3) of the robot's global location in space x_R, y_R and global orientation θ_R

$$x_R^{t+1} = x_R^t + v^t cos(\theta^t) \quad (4)$$
$$y_R^{t+1} = y_R^t + v^t sin(\theta^t) \quad (5)$$
$$\theta^{t+1} = \theta^t + \omega^t \cdot dt \ . \quad (6)$$

For the ball, motion is propagated by a constant velocity model with a simple exponential decay on the initial ball velocity $\dot{x}^0_B, \dot{y}^0_B$ to mimic frictional loss,

$$\dot{x}_B^{t+1} = \alpha \dot{x}_B^t \qquad \dot{y}_B^{t+1} = \alpha \dot{y}_B^t \quad (7)$$
$$x_B^{t+1} = x_B^t + \alpha^t \dot{x}_B^t dt \qquad y_B^{t+1} = y_B^t + \alpha^t \dot{y}_B^t dt \quad (8)$$

where $\alpha \in [0, 1]$ is the decay constant. These positions generated within the domain are bounded above and below, with both the ball and the robot being constrained to remain within these bounds. We assume there are dynamic limitations on the performance of the robot whereby its linear and rotational accelerations are limited to $[-\dot{v}_{max}, \dot{v}_{max}]$ and $[-\dot{\omega}_{max}, \dot{\omega}_{max}]$, and its linear and rotational velocity bounded within $[-v_{max}, v_{max}]$ and $[-\omega_{max}, \omega_{max}]$, respectively.

5.3 Performance Evaluation

To measure the performance of our algorithm, trajectory executions were evaluated for success and efficiency. Since these measures were used by our teacher when forming a critique, it was along such criterion that we expected to see learner performance improve. 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.

The teacher demonstrates during the training phase from initial world conditions in the set I_{train}, and the learner during the critiquing phase from those in the set I_{critiq}. We define a *critiquing round* as execution from a single initial condition in I_{critiq}.

To evaluate the benefit of critiquing, we introduce a third *testing phase*, within initial conditions I_{test}. This phase was carried out after every n critiquing rounds, for evaluation purposes only. For our implementation, $n = 20$. The learner would execute the most recent update on its policy π, but no teacher critiquing would occur. Executions were evaluated for interception success and efficiency, as defined above.

5.4 Algorithm Application

We now present the specifics of applying our algorithm to the stated problem domain. The teacher interacts in two different phases of our system. During the critiquing phase, the teacher was a human. Since this intial work was done in simulation, during the training phase the teacher was a hand-written control policy. Our intent is that when applying this algorithm to our real robot, teacher interaction during the training phase will also be performed by a human.

The hand-written control policy which guided our teacher's motion was a set of simple rules to adjust rotational and translational speed. We treat this control policy as a black box. It is worthwhile to note is that this control policy is not optimal for the domain, and does not always successfully intercept the ball. This is because it was hand coded by a human without knowledge of what that optimal solution might be, and our goal was not to optimize this policy, but rather to investigate the knowledge transfer between teacher and robot.

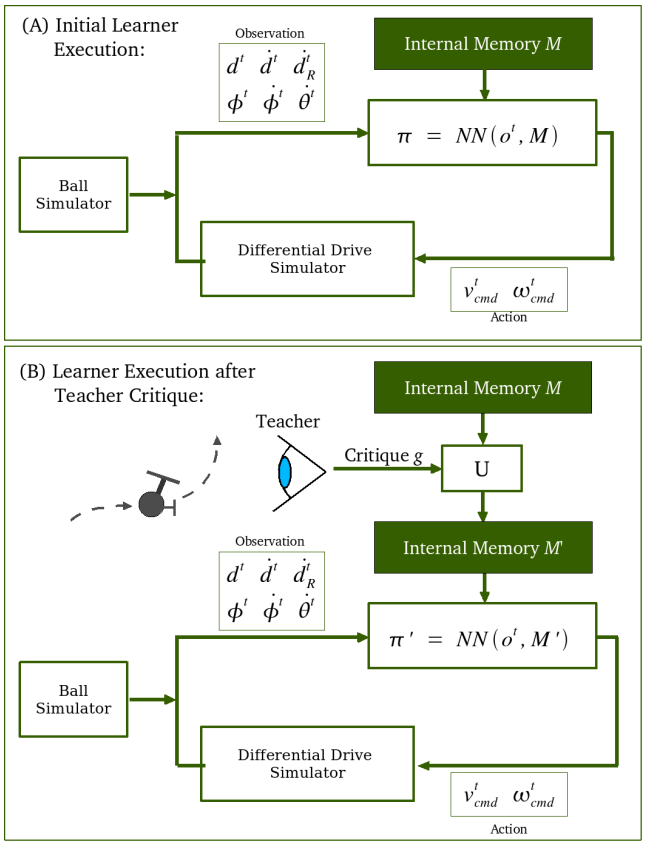

Figure 5: Single time step from an execution-critique-update cycle. (A) The learner executes policy π from M. (B) This execution is critiqued by the teacher, the learner updates $M' \leftarrow M$ and then executes an updated policy π' from M'.

A single execution-critique-update cycle within the critiquing phase is shown in Figure 5. During our nearest neighbor calculations, the distance metric used was Euclidean,

$$D(o^t, o_i^j, m_i^j) = m_i^j \left(o^t - o_i^j\right)^T \Sigma^{-1} \left(o^t - o_i^j\right). \quad (9)$$

Here Σ is a diagonal matrix which encodes the range of each observation dimension, and thus removes the effect of different units in each dimension. The factor m_i^j scales the entire distance calculation based upon the past recommendation success of observation point o_i^j. In its update (Eq.3), $\kappa = 0.1$ was set based upon typical 'close' distances within the domain. Additionallly, the amount by which m_i^j could

increase between time steps was bounded above by 1. Thus the m_i^j of a poorly performing data point was increased by the maximum amount of 1 if located within distance κ of the query point, and by an amount less than 1 (which decayed exponentially with distance) otherwise.

The physical dimensions of the simulated world correspond to a Segway scale and were 5.0m x 10.0m. The world was robot-centric. At each execution run start the robot was located at the center of the world $(0, 0)$, and oriented towards $(1, 0)$. Due to symmetry, a control policy developed for one 5.0m x 5.0m quadrant easily translates into a control policy for the full 5.0m x 10.0m world, by simply flipping the sign on all x locations and rotational velocity commands ω^t. We operated within the quadrant of positive x, y positions, with $(0, 0)$ located in the lower left corner.

The acceleration and velocity constraints which were set for the simulated robot are comparable for our real differential drive robot (Table 1). Ball speeds were also set to be bounded such that $\|\dot{x}_B^t, \dot{y}_B^t\| < 2.5m/s$ and decayed according to Eq.7 with $\alpha = 0.99$.

$\dot{v}_{max}$	3.0 m/s^2	$\dot{\omega}_{max}$	7.0 rad/s^2
v_{max}	2.0 m/s	ω_{max}	3.0 rad/s

Table 1: Acceleration and Velocity Constraints

Each initial condition consisted of initial ball position and speed, and robot state. Three distinct sets of initial conditions existed for each phase, so that $I_{test} \neq I_{critiq} \neq I_{train}$. They were generated from a uniform subsampling of S, within set ranges. I_{test} and I_{critiq} were sampled continuously, and I_{train} from a grid. The size of each set was 100, 120 and 127, respectively. The ranges of the subsampling were: $x_B^0, y_B^0 \in [0.0, 5.0]$ and $\dot{x}_B^0, \dot{y}_B^0 \in [-2.5, 0.0]$. Note that within such ranges, the ball is constrained to travel initially towards the robot. The initial conditions were additionally pruned for feasiblity with simple checks, such as whether the robot travelling at top speed could even reach the location where the ball would go out of bounds before the ball did.

6. RESULTS

Now we present the results of applying our algorithm to the stated problem domain. In particular, 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 6.

6.1 More Successful Executions

Learner performance was found to improve with teacher critiquing. This improvement was seen both within individual subsets of the critiquing phase, as well as under validation by an independent test set.

Testing with the independent test set was performed intermittently throughout the critiquing phase, to mark learner progress. Figure 7 shows learner improvement, where each datapoint represents the result of executing from all conditions within I_{test}. The learner begins executions using

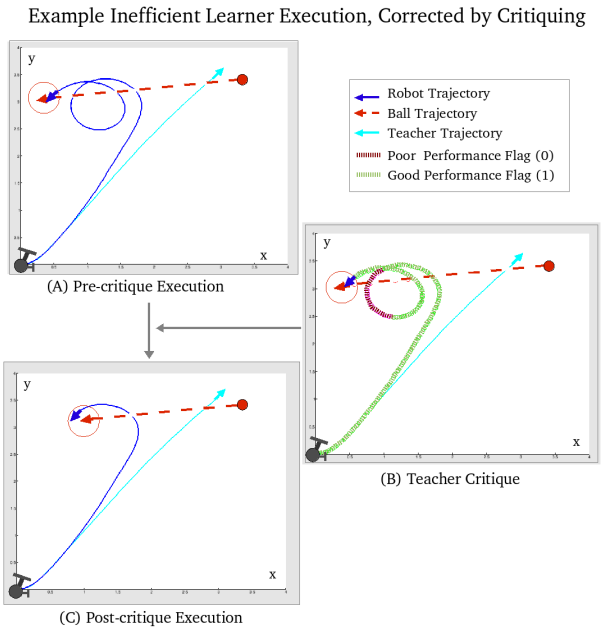

Figure 6: Example learner execution made more efficient by critiquing. (A) The robot initially intercepts the ball, but the loop in its trajectory is inefficient. (B) The teacher critiques this trajectory, flagging the loop as poor. (C) The robot repeats the execution successfully without a loop. Arrow heads indicate direction of travel, and the red circle the distance threshold for successful interception. Note that the hand-written control policy of our teacher does not successfully intercept the ball.

the initial internal memory M derived from TR (Initial M), which it updates after each critiquing round. After the final critiquing round, a final version of M exists (Final M). Learner percent success, using the initial and final versions of M, are shown in Table 2. The percent improvement on learner performance (Table 3) on the test set was 25.00% (calculated as the difference in the number of successes using Inital M and Final M, over the number of successes using Initial M). For comparison, these same initial conditions were tested with the teacher's hand-written control policy. The learner's performance improved upon the teacher's by 12.90%. To check performance with the set upon which the learner was critiqued, the same results were gathered with initial conditions I_{critiq}, with similar results.

Condition Set	Learner, Initial M	Learner, Final M	Teacher
I_{critiq}	55.00%	71.67%	61.67%
I_{test}	**56.00%**	**70.00%**	62.00%

Table 2: Execution Percent Success

Condition Set	Impr over Self	Impr over Teacher
I_{critiq}	30.31%	16.22%
I_{test}	**25.00%**	12.90%

Table 3: Learner Percent Improvement in Success

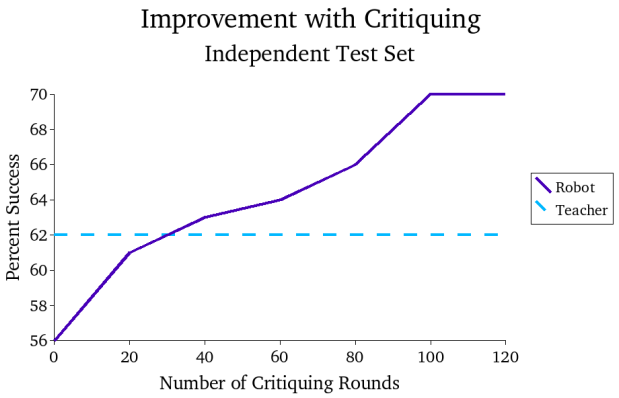

Figure 7: Improvement in learner performance on the test set with critiquing rounds (solid line). (Hand-written control policy test set performance provided for comparison, dashed line).

That the learner was able to perform better than the hand-coded control policy underlines the benefits of critiquing within this domain. The hand-coded policy is not optimal for the domain. By critiquing our robot's executions, we were able to correct for some demonstration error and improve the robot's performance beyond the capabilities of the demonstration control policy, and all in a simple and straightforward manner.

6.2 More Efficient Executions

Learner executions were found to become more efficient with critiquing. That is, the robot intercepts the ball faster, indicated by a reduction in trajectory length. In particular, we compared the average lengths of learner execution with the initial version of M, learner execution with the final version of M, and teacher execution (Table 4). With the independent test set, critiquing enabled the learner to reduce the length of its executions by 27.98% (Table 5). Note that we consider only scenarios in which both learner and teacher are initially successful, since for the same initial ball position and velocity a successful interception is always faster, and therefore has a shorter trajecotry length, than an unsuccessful one.

Condition Set	Learner, Initial M	Learner, Final M	Teacher
I_{critiq}	2.94	2.03	2.97
I_{test}	**2.73**	**1.96**	2.87

Table 4: Average Execution Length (seconds)

Condition Set	Impr over Self	Impr over Teacher
I_{critiq}	44.94%	31.59%
I_{test}	**27.98%**	31.71%

Table 5: Percent Reduction in Execution Length

The criteria by which the teacher critiqued learner performance 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. 6). To formally define a metric for credit assignment which determines 'wrongness' and 'smartness', however, would be quite difficult. It is exactly in these sort of intuitive judgement calls when having the critiquing teacher be *human* becomes so important to the system.

6.3 Retention of Initial Critique Success

In the critiquing phase, M is updated after each critiquing round. For most past executions, we would expect this new information to improve learner performance. For some, however, it is possible that performance might decline.

When updating m_i^j (Eq. 3), our metric does not take into consideration *where* query points are in relation to training data points. Rather, only the *distance* between is considered. Consider two query points located at identical distances to, but in orthogonal directions from, a given training point. The training point's recommended action might be appropriate for one query point but not the other, and so its execution would incur different successes, and therefore also different critiques, for each. By such reasoning, it is possible in general that trajectory executions may incur critiques which effectively reverse the critiques of earlier executions, or penalize recommendations which actually are appropriate under different conditions. Therefore, a trajectory which executed successfully under a prior version of M may possibly perform poorly under a later version of M. Consequently, whether a critique was *initially* successful is not necessarily captured by whether the trajectory executes successfully under the final version of M. The incorporation of query point orientation into the update of m_i^j is thus a potential improvement for this algorithm.

To investigate the initial success of a given critique, we first looked at a trajectory's execution success before receiving its critique, using internal memory M. We then compared this to the success of the repeat execution immediately following the critiquing round, using the modified M'. Table 6 presents a summary of initial execution successes (in groups of 20 averaged critiquing rounds, for compactness). Indeed, the average post-critique, modified M' success of 74.17% exceeds 71.67% (Table 2), the average percent success of executing I_{critiq} with the final version of M that results from completing all critiquing rounds. This indicates that not all successful critiques are properly retained within M, though we take note that a net result of improved performance is in any case still observed.

Round	Starting M, pre-critique	Modified M', post-critique
1-20	65.00%	80.00%
21-40	60.00%	75.00%
41-60	50.00%	55.00%
61-80	50.00%	60.00%
81-100	65.00%	90.00%
101-120	75.00%	85.00%
Average	60.83%	74.17%

Table 6: Pre- and Post-Critique Percent Success

6.4 Reduction in Effective Critiquing

Our results show a reduction in effective teacher critiquing as the robot executes within the environment. By effective critiquing, we mean a trajectory which successfully intercepts the ball following a critique from the teacher. Note that this decrease was paralleled by an overall increase in robot performance on the test set.

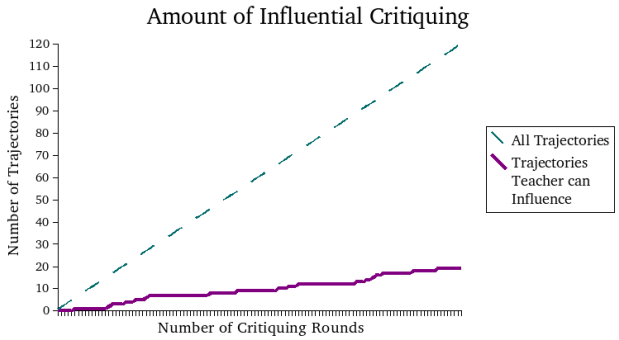

Figure 8: Number of critiquing rounds which resulted in successful interceptions (solid line). (Total number of critiquing rounds shown for reference, dashed line).

In total, 36/120 of the trajectories executed from the initial conditions of I_{critiq} were chosen by the teacher for critique. After critiquing a trajectory, the teacher would ask the learner to repeat execution, to verify that actions which had been flagged as poor were corrected. On average, the learner was asked to repeat a critiqued execution 2.61 times. Repetitions were no longer requested once the teacher determined a successful trajectory (a) had been satisfactorily executed, (b) was infeasible for the robot given world constraints, or (c) was infeasible given the underlying training data. Note that the later two reasons might be cause for the teacher to never critique a trajectory in the first place, if believing successful execution to be infeasible for the robot.

In particular, we identify two distinct sources for learner execution error. First are errors which result from learner *application* of the underlying training data, which may be corrected by our algorithm and the critique of the teacher. Second are errors that result from an *inadequacy* in training data or physical capability, which cannot be corrected by our algorithm. As this first form of error reduces, so do the options for our teacher to provide effective critiques within the domain. Indeed, the number of effectively critiqued trajectories did not increase linearly with the number of executed trajectories (Fig. 8).

7. CONCLUSIONS

We contribute an algorithm for learning a robot's control policy, which makes use of both the demonstrations and critique of a teacher. We have argued that this critiquing task is well-suited for a human teacher, and simpler than hand-coding a control policy. We validate the algorithm with a first application in simulation, and it produces an effective control policy for the domain. The results of our simulation application demonstrate that the critique of a human teacher does improve task performance, where task performance is measured by both execution success and efficiency. On both measures learner performance not only improved, but came to exceed the performance of its teacher. Future work will focus on implementing this algorithm on a SegwayRMP robots performing a similar task.

8. REFERENCES

[1] A. Alissandrakis, C. L. Nehaniv, and K. Dautenhahn. Imitation with ALICE: Learning to imitate corresponding actions across dissimilar embodiments. *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, 32(4), July 2002.

[2] C. G. Atkeson, A. W. Moore, and S. Schaal. Locally weighted learning for control. *Artificial Intelligence Review*, 11:75–113, 1997.

[3] C. G. Atkeson and S. Schaal. Robot learning from demonstration. In J. Douglas H. Fisher, editor, *Machine Learning: Proceedings of the Fourteenth International Conference (ICML '97)*, pages 12–20, San Francisco, CA, 1997.

[4] J. Bagnell, S. Kakade, A. Ng, and J. Schneider. Policy search by dynamic programming. In *Neural Information Processing Systems*, volume 16. MIT Press, December 2003.

[5] D. C. Bentivegna. *Learning from Observation using Primitives*. PhD thesis, Georgia Institute of Technology, 2004.

[6] A. Billard and M. Mataric. Learning human arm movements by imitation: Evaluation of biologically inspired connectionist architecture. *Robotics and Autonomous Systems*, 37(2-3):145–160, November 30 2001.

[7] G. Hayes and J. Demiris. A robot controller using learning by imitation. In *Proceedings of the 2nd International Symposium on Intelligent Robotic Systems*, Grenoble, France, July 1994.

[8] A. Ijspeert, J. Nakanishi, and S. Schaal. Movement imitation with nonlinear dynamical systems in humanoid robots. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA2002)*, pages 1398–1403, 2002.

[9] M. J. Mataric. Learning in behavior-based multi-robot systems: Policies, models, and other agents. *Cognitive Systems Research, Special Issue on Multi-disciplinary studies of multi-agent learning*, 2(1):81–93, 2001.

[10] H. Miyamoto and M. Kawato. A tennis serve and upswing learning robot based on bi-directional theory. *Neural Networks*, Special Issue(11):1331–1344, 1998.

[11] M. N. Nicolescu and M. J. Mataric. Methods for robot task learning: Demonstrations, generalization and practice. In *Second International Joint Conference on Autonomous Agents and Multi-Agent Systems*, Melbourne, Australia, July 2003.

[12] B. Robins, K. Dautenhahn, R. te Boekhorst, and A. Billard. Effects of repeated exposure to a humanoid robot on children with autism. In S. Keates, J. Clarkson, P. Langdon, and P. Robinson, editors, *Designing a More Inclusive World*, pages 225–236. Springer-Verlag Press, London, March 2004.

[13] A. Ude, C. G. Atkeson, and M. Riley. Programming full-body movements for humanoid robots by observation. *Robotics and Autonomous Systems*, 47:93–108, 2004.