

Mobile Robot Motion Control from Demonstration and Corrective Feedback

Brenna D. Argall, Brett Browning, and Manuela M. Veloso

Abstract Robust motion control algorithms are fundamental to the successful, autonomous operation of mobile robots. Motion control is known to be a difficult problem, and is often dictated by a *policy*, or state-action mapping. In this chapter, we present an approach for the refinement of mobile robot motion control policies, that incorporates *corrective* feedback from a human teacher. The target application domain of this work is the low-level motion control of a mobile robot. Within such domains, the rapid sampling rate and continuous action space of policies are both key challenges to providing policy corrections. To address these challenges, we contribute *advice-operators* as a corrective feedback form suitable for providing *continuous*-valued corrections, and *Focused Feedback For Mobile Robot Policies (F3MRP)* as a framework suitable for providing feedback on policies sampled at a high frequency. Under our approach, policies refined through teacher feedback are initially derived using *Learning from Demonstration (LfD)* techniques, which generalize a policy from example task executions by a teacher. We apply our techniques within the *Advice-Operator Policy Improvement (A-OPI)* algorithm, validated on a Segway RMP robot within a motion control domain. A-OPI refines LfD policies by correcting policy performance via advice-operators and F3MRP. Within our validation domain, policy performance is found to improve with corrective teacher feedback, and moreover to be similar or superior to that of policies provided with more teacher demonstrations.

Brenna D. Argall
Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, USA e-mail: bargall@ri.cmu.edu

Brett Browning
Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, USA e-mail: brettb@cs.cmu.edu

Manuela M. Veloso
Computer Science Department, Carnegie Mellon University, Pittsburgh, PA, USA e-mail: veloso@cs.cmu.edu

1 Introduction

Whether an exploration rover in space or recreational robot for the home, successful autonomous mobile robot operation requires an algorithm for motion control. A control *policy* provides such an algorithm, by mapping an observation of the world to an action available on the robot. This mapping is fundamental to many robotics applications, yet in general is complex to develop.

Even with a carefully crafted policy, a robot often will not behave as the developer expects or intends in all areas of the execution space. One way to address behavior shortcomings is to update a policy based on execution experience, which can increase policy robustness and overall performance. For example, such an update may expand the state-space areas in which the policy is valid, or increase the likelihood of successful task completion.

This chapter contributes our approach for refining mobile robot policies with human feedback. The feedback consists of policy corrections, which are provided based on human teacher observations of policy executions by the robot. The target domain of this work is low-level motion control on a mobile robot. Challenges to providing corrective feedback within this domain include the continuous state-action space of the policy, and the rapid rate at which the policy is sampled. We introduce techniques to address both of these challenges.

1.1 Mobile Robot Motion Control

A motion control policy defines a mapping from world state to robot action. Motion control policies are able to represent actions at a variety of control levels:

Low-level actions: Low-level actions directly control the movement mechanisms of the robot. These actions are in general continuous-valued and of short time duration, and a low-level motion policy is sampled at a high frequency. An example low-level action is a command for wheel speed that updates at $50Hz$.

High-level actions: High-level actions encode a more abstract action representation, which is then translated through other means to affect the movement mechanisms of the robot; for example, through another controller. These actions are in general discrete-valued and of longer time duration, and their associated control policies are sampled at a low frequency. An example high-level action is to approach and pick up an object, executing over tens of seconds.

The focus of this chapter is on low-level motion control policies. The continuous action-space and high sampling rate of low-level control are both key considerations during policy development and refinement.

The state-action mapping represented by a motion control policy is typically complex to develop. One reason for this complexity is that the target observation-action mapping is unknown. What *is* known is the desired robot motion *behavior*,

which must somehow be represented through an unknown observation-action mapping. A second reason for this complexity are the complications of motion policy execution in real world environments. The world is observed through sensors that are typically noisy; models of world dynamics are only an approximation to the true dynamics; and actions are motions executed with real hardware, which depends on physical considerations like calibration accuracy and inevitably executes with some level of imprecision.

The development of control policies therefore generally requires a significant measure of effort and expertise, and frequently demands extensive prior knowledge and parameter tuning. The required prior knowledge ranges from details of the robot and its movement mechanisms, to details of the execution domain and how to implement a given control algorithm. Any successful application typically has the algorithm highly tuned for operation with a particular robot in a specific domain. Furthermore, existing approaches are often applicable only to simple tasks due to computation or task representation constraints.

Traditional approaches to robot control model the domain dynamics and derive policies using those mathematical models [35]. While theoretically well-founded, these approaches typically depend heavily upon the accuracy of the model, which can require considerable expertise to develop and becomes increasingly difficult to define as robot become more complex. Other approaches, such as *Reinforcement Learning (RL)* [37], guide policy learning by providing reward feedback about the desirability of visiting particular states. To define a function to provide these rewards, however, is known to be a difficult problem that also requires considerable expertise to address. Furthermore, building the policy necessitates gathering information by visiting states to receive rewards, which is non-trivial for a mobile robot executing physical actions.

The experience of personally developing numerous motion behaviors by hand for this robot [5], and subsequent desire for more straightforward policy development techniques, was a strong motivating factor in this work. Similar frustrations have been observed in other roboticists, further underlining the value of approaches that ease the policy development process. Another, more hypothetical, motivating factor is that as familiarity with robots within general society becomes more prevalent, it is expected that future robot operators will include those who are *not* robotics experts. We anticipate a future requirement for policy development approaches that not only ease the development process for experts, but are accessible to non-experts as well.

1.2 Learning from Demonstration

Learning from Demonstration (LfD) is one policy development technique with the potential for both application to non-trivial tasks and straightforward use by robotics-experts and non-experts alike [4, 11]. Under the LfD paradigm, a teacher first demonstrates a desired behavior to the robot, producing an example state-action

trace. The robot then generalizes from these examples to derive a policy, thus learning a state-action mapping.

1.2.1 Support for Demonstration Learning

LfD has many attractive points for both learner and teacher. To develop a policy within a LfD paradigm typically does not require expert knowledge of the domain dynamics, which removes the performance dependence on model accuracy. The relaxation of the expert knowledge requirement also opens policy development to those who are not robotics-experts, satisfying a need that we expect to increase as robots become more commonly available. Furthermore, demonstration has the attractive feature of being an intuitive medium for communication from humans, who already use demonstration to teach other humans.

More concretely, the application of LfD to motion control has many advantages:

Implicit behavior to mapping translation. By demonstrating a desired motion behavior, and recording the encountered states and actions, the translation of a behavior into a representative state-action mapping is immediate and implicit.

Robustness under real world uncertainty. The uncertainty of the real world means that multiple demonstrations of the same behavior will not execute identically. Generalization over demonstration examples thus produces a policy that does not depend on a strictly deterministic world, and therefore should execute more robustly under real world uncertainty.

Focused policies. Demonstration has the practical feature of focusing the dataset of examples to areas of the state-action space actually encountered during behavior execution. This is particularly useful in continuous action space domains, with an infinite number of state-action combinations.

The LfD approach to obtaining a policy is in contrast to other techniques in which a policy is learned from *experience*, for example building a policy based on data acquired through exploration, as in RL. Furthermore a policy derived under LfD is necessarily defined only in those states encountered, and for those actions taken, during the example executions.

1.2.2 Formalism

Our approach to policy development derives an initial policy from teacher demonstrations. Within this chapter, we formally define the world to consist of states S and actions A , with the mapping between states by way of actions being governed 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, and instead the learner has access to observed state Z , through a mapping $S \rightarrow Z$. A teacher demonstration $d \in D$ is represented as n pairs of observations and actions, such that $d = \{(\mathbf{z}_i, \mathbf{a}_i)\} \in D$, $\mathbf{z}_i \in Z$, $\mathbf{a}_i \in A$, $i = 0 \dots n$.

Within the typical LfD paradigm, the set D of these demonstrations is provided to the learner. 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 the dataset D .

1.2.3 Related Work

LfD has found success on a variety of robot platforms and applications [4, 11]. Since demonstration for real robots involves executing actions in physical environments, differences in *embodiment* between the learner and teacher become of crucial importance. The challenges that arise from these differences, where teacher demonstrations may not map directly to the learner due to differences in sensing or motion, are broadly referred to as *correspondence issues* within the LfD literature [12, 26]. Key design decisions therefore include the choice of teacher controlling, and platform executing, the demonstration, as well as how the demonstration is recorded.

A variety of approaches exist for executing and recording teacher demonstrations. At one extreme lies *teleoperation*, where the passive robot platform records from its own sensors while under direct teacher control [9, 13]. This approach is very effective at reducing teacher-learner correspondence issues, but does require actively controlling the robot during the task, which might not be manageable for example if controlling a high-DoF humanoid for low-level motion control. Another approach has the robot learner actively *mimic* the teacher during the demonstration executions, again while recording from its own sensors [18, 29]. This has the advantage of not requiring the teacher to actively control the robot, but does require that the learner be able to identify and track the teacher; furthermore, the observations made by the *teacher* during the execution are not directly recorded.

Other demonstration techniques do not employ the actual learner platform during the demonstration. One such technique has the *teacher* wear sensors during task execution with her *own* body [14, 21, 23]. This requires specialized sensors and introduces another level of teacher-learner correspondence, but does not require that the learner platform be actively operated or able to track the teacher during task execution. Lastly, at the opposite extreme to teleoperation, sensors *external* to the teacher's body may record his execution of the task with his own body [8, 10]. This has the lowest requirements in terms of specialized sensors or actively operating the robot, but is the most likely to encounter correspondence issues when transferring the recorded teacher demonstrations to the learner platform.

Once the dataset is recorded, a policy must be derived from it. A variety of policy derivation techniques are employed within LfD, the majority of which fall into three categories. The first category directly *approximates the function mapping* states to actions $f() : Z \rightarrow A$, using regression or classification techniques. Successful LfD implementations of this policy derivation approach include tasks with humanoids [10, 14, 20], Sony AIBOs [15, 19] and a variety of other platforms [25]. The second category learns a *state-action transition model* $T(s'|s, a)$ from the demonstration data, pairs this with a reward function $R(s)$ (either hand-engineered or learned) and derives a policy using RL techniques. Successful LfD applications under this ap-

proach span tasks with autonomous helicopters [1, 9, 27] to small quadriped robots [22, 32], humanoids [24] and robotic arms [8]. The third category learns task *plans* from the demonstration set, including small wheeled robot [29] and companion robot [33] applications. Note that each of these techniques are employed for the derivation of policies with high-level actions, while only the first two are used to derive policies with low-level actions.

Our work takes the policy derivation approach of directly approximating the underlying function mapping states to actions. Since our action space is continuous, regression techniques are employed. During the empirical validation of our techniques, teleoperation is the approach that will be used for gathering demonstrations.

1.3 Policy Refinement within LfD

LfD is inherently linked to the information provided in the demonstration dataset. As a result, learner performance is heavily limited by the quality of this information. Though LfD has enabled successful policy development for a variety of robot platforms and applications, this approach is not without its limitations.

1.3.1 Potential Dataset Limitations

One common cause for poor learner performance is dataset sparsity, or the existence of state space areas in which no demonstration has been provided. Dataset sparsity is a trade off to focusing the dataset to areas visited during task execution, since the learner is provided with an indication of which action to take only in those states visited during demonstration. In all but the most simple domains the teacher will be unable to demonstrate from every state, and so there will be areas of the state space absent from the demonstration set. Note however that dataset sparsity may be overcome to a certain extent by the generalization ability of the policy derivation technique.

A second cause is poor quality of the dataset examples. Poor quality examples can result from the demonstration abilities of the teacher, who may in fact provide suboptimal or ambiguous demonstrations. Poor quality examples also can result from poor correspondence between the teacher and learner, who may differ in sensing or motion capabilities.

To summarize, common sources of LfD limitations include:

1. Uncovered areas of the state space, absent from the demonstration dataset.
2. Suboptimal or ambiguous teacher demonstrations.
3. Poor translation from teacher to learner, due to correspondence issues.

One way to address dataset limitations is to extend LfD by having the robot update its policy based on execution experience.

1.3.2 Related Work

One popular approach for dealing with poor or ambiguous teacher demonstrations is to provide more demonstration data in response to execution experience with the policy. There are approaches that acquire new demonstrations by enabling the learner to evaluate its confidence in selecting a particular action, based on the confidence of the underlying classification algorithm. For example, the robot can indicate to the teacher its certainty in performing various elements of the task [19], or request additional demonstration in states that are either very different from previously demonstrated states or for which a single action cannot be selected with certainty [16]. Other approaches rely on teacher observation of the policy performance alone, for example to provide a robot with new demonstrations through kinesthetic teaching that moves passive joints through desired position trajectories [14].

Another approach is to pair LfD with RL techniques, which is particularly relevant for implementations that already derive their policies using RL. The goal of RL is to maximize cumulative reward over time, and typically each state s is associated with a value according to the function $V(s)$ (or associating state-action pair s, a with a Q-value according to the $Q(s, a)$) [37]. By updating the state values $V(s)$ with rewards received during execution [36], a policy derived under LfD also updates. We note that these are rewards seen during *learner* execution, and not during demonstration. To visit and evaluate new states not seen in the demonstration set, an exploration policy may be employed [30, 34], though we note that in general taking exploratory steps on a real robot can be inefficient and even dangerous. Finally, execution experience may also update a learned transition function $T(s'|s, a)$ [2].

2 Corrective Feedback for Policy Refinement

Our approach to the improvement of LfD policies through experience is to provide corrections on policy executions. Corrective feedback is provided by a human teacher, in response to policy executions by a robot learner. In particular, feedback corrects state-action mappings produced during a student execution to generate new examples for the LfD policy. We begin by motivating and discussing corrective feedback for the improvement of LfD policies (Sec. 2.1), followed by a detailing of the contributed techniques that enable a human teacher to provide continuous-valued corrections (Sec. 2.2) to policies sampled at a high frequency (Sec. 2.3). We then present an algorithm that employs our corrective feedback techniques (Sec. 2.4).

2.1 Policy Corrections

To address potential LfD limitations, the approach of correcting poor policy predictions we argue is particularly direct. While overall performance evaluations or state

rewards can provide an indication of the quality of a policy prediction, they do not provide any guidance on what might have been a more suitable *alternate prediction*. Providing a correction on poor predictions therefore provides more focused and detailed policy improvement information. Furthermore, while more demonstrations can populate sparse areas of the state space or demonstrate a corrected behavior, they require *state re-visitation*, which can be impractical within real world domains.

Policy correction has seen limited attention within LfD however. The selection of a policy correction in general is sufficiently complex to preclude it being provided with a simple function. Approaches that do correct policy predictions therefore provide corrections through human teachers, which our techniques do as well. Furthermore, since the correction involves selecting a preferred state or action, within the existing literature corrections are only provided within action spaces that are *discrete* and with actions of significant time duration, and therefore sampled with *low* frequency. For example, the correct action from a discrete set is provided by a human teacher to update a high-level action classifier [15], and the structure of a hierarchical Neural Network of robot behaviors [29].

Approaches that correct policies within *continuous* action-spaces sampled at *high* frequency are absent from the existing LfD literature. These considerations have prompted our development of a corrective feedback form that *is* appropriate for continuous-valued action domains (Sec. 2.2). We furthermore develop a feedback framework (Sec. 2.3) that is suitable for domains with rapidly sampled policies.

2.2 Advice-Operators

To address the challenge of providing continuous-valued corrections, we introduce *advice-operators* [3] as a language through which a human teacher provides policy corrections to a robot student.

Concretely defined, an advice-operator is a mathematical computation performed on an observation input or action output. Given a policy execution by the learner, an operator is indicated by the teacher and applied to a state-action pair recorded during the execution. Key characteristics of advice-operators are that they:

1. Perform mathematical computations on datapoints.
2. Are defined commonly between the student and advisor.
3. May be applied to observations or actions.

Figure 1 presents a diagram of data synthesis from student executions and teacher feedback (bottom, shaded area); for comparison, LfD data from teacher executions is also shown (top). To illustrate with an example, consider a simple operator that modifies translational acceleration by a static amount δ . Suppose the teacher indicates this operator for application over 15 data points from the learner execution. The translational speed a^0 of executed point 0 then updates to $\hat{a}^0 \leftarrow a^0 + \delta$, point 1 to $\hat{a}^1 \leftarrow a^1 + \delta$, and so forth until point 14 updates to $\hat{a}^{14} \leftarrow a^{14} + \delta$.

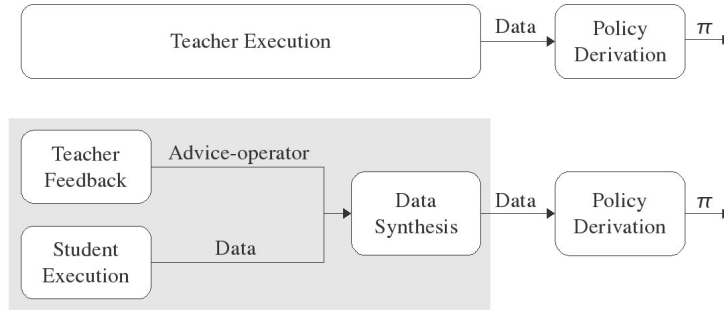


Fig. 1 Generating demonstration data under classical LfD (top) and advice-operators (bottom).

Policy correction under advice-operators does *not* rely on teacher demonstration to indicate a corrected behavior. The advice-operator approach thus includes the following strengths:

No need to recreate state. This is especially useful if the world states where corrective demonstration is needed are dangerous (e.g., lead to a collision), or difficult to access (e.g., in the middle of a motion trajectory).

Not limited by the demonstrator. Corrections are not limited to the execution abilities of the demonstration teacher, who may be suboptimal.

Unconstrained by correspondence. Corrections are not constrained by physical differences between the teacher and learner.

Possible when demonstration is not. Further demonstration may in fact be impossible (e.g., teleoperation over a 40 minute Earth-Mars communications lag).

We thus contribute a formulation for corrective feedback, as a predefined list of mathematical functions. Advice-operators enable the translation of a statically-defined high-level correction into a continuous-valued, execution-dependent, low-level correction. Moreover, when combined with our techniques for *providing* feedback (Sec. 2.3), a single piece of advice corrects multiple execution points. The selection of a *single advice-operator* thus translates into *multiple continuous-valued corrections*, and therefore is suitable for modifying low-level motion control policies sampled at high frequency.

2.3 Focused Feedback for Mobile Robot Policies

To address the challenge of providing feedback to policies sampled at a rapid rate, we introduce *Focused Feedback for Mobile Robot Policies (F3MRP)* [6] as a framework through which portions of a policy execution are selected to receive feedback. The target application domain for F3MRP is *mobile* robot motion control.

At the core of the F3MRP framework is a visual presentation of the 2-D path physically taken by the mobile robot on the ground.¹ For experiments with a simulated robot, the 2-D path is represented in real-time as the robot executes. For experiments with a real robot, the 2-D path is played back, at true speed, after the learner execution completes, to mitigate inaccuracies due to network lag.

The visual path presentation is a key component of the interface for the identification of those portions of the learner execution that require correction. Through this interface, the teacher selects segments of the 2-D ground path that correspond to those portions of the execution during which the policy performed poorly. An overview of the F3MRP interface is shown in Figure 2, which expands the shaded area in Figure 1 with the details of segment selection.

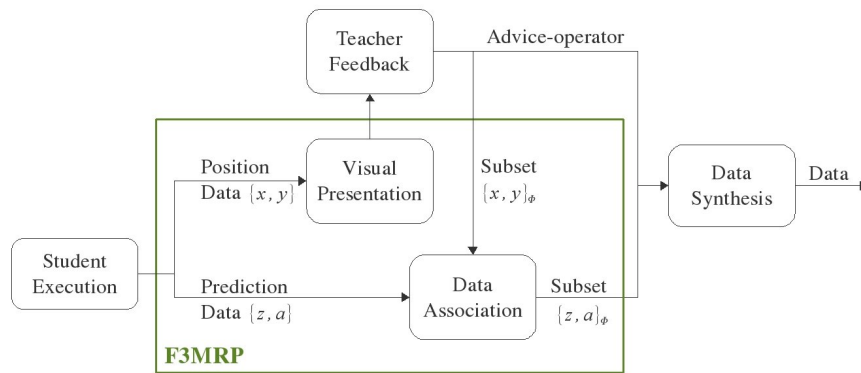


Fig. 2 Path visualization, subset selection and data association under the F3MRP framework.

Corrective feedback provided under the F3MRP framework must associate closely with the underlying learner execution, since the feedback corrects specific execution points. By contrast, consider an overall performance measure, that need only associate with the execution as a whole and thus links to the data at a fairly coarse scale. To accomplish close feedback-execution association under F3MRP, the teacher selects problem segments of the graphically displayed ground path. Segment sizes are determined dynamically by the teacher, and may range from a single point to all points in the trajectory.

The F3MRP framework then associates the selected segment of the *position trace*, i.e. the ground path, with the corresponding segment of the *prediction trace*, i.e. the state-action sequence, recorded during the learner execution. This process is the tool through which the human flags state-action pairs for modification: by selecting segments of the displayed ground path, which the framework then associates with the state-action trace of the policy.

¹ The F3MRP framework was designed specifically for mobile robot applications. To apply the framework to non-mobile robots would require an alternative to the 2-D ground path, to serve as the visualization component of the interface for segment selection.

2.4 Algorithm Advice-Operator Policy Improvement

The *Advice-Operator Policy Improvement (A-OPI)* algorithm [3] refines a motion control policy, initially derived from LfD, by providing corrections through advice-operators and the F3MRP framework. The algorithm operates in two phases. During the *demonstration phase*, a set of teacher demonstrations is provided to the learner. This demonstration set D consists of example executions of the target behavior, during which state-action pairs are recorded. From this set the learner generalizes an initial policy. During the *feedback phase*, the learner executes with this initial policy. Feedback on the learner execution is offered by a human teacher, and is used by the learner to update its policy. The learner then executes with the updated policy, and the *execute-feedback-update* cycle continues to the satisfaction of the teacher.

Algorithm 1 Advice-Operator Policy Improvement

```

1: Given  $D$ 
2: initialize  $\pi \leftarrow \text{policyDerivation}(D)$ 
3: while practicing do
4:   initialize  $d \leftarrow \{\}, tr \leftarrow \{\}$ 
5:   repeat
6:     predict  $\mathbf{a}^t \leftarrow \pi(\mathbf{z}^t)$ 
7:     execute  $\mathbf{a}^t$ 
8:     record  $d \leftarrow d \cup (\mathbf{z}^t, \mathbf{a}^t)$ ,  $tr \leftarrow tr \cup (x^t, y^t, \theta^t)$ 
9:   until done
10:  advise  $\{op, \Phi\} \leftarrow \text{teacherFeedback}(tr)$ 
11:  for all  $\varphi \in \Phi$ ,  $(\mathbf{z}^\varphi, \mathbf{a}^\varphi) \in d$  do
12:    if  $op$  is observation-modifying then
13:      modify  $(\mathbf{z}^\varphi, \mathbf{a}^\varphi) \leftarrow (op(\mathbf{z}^\varphi), \mathbf{a}^\varphi)$ 
14:    else  $\{op$  is action-modifying $\}$ 
15:      modify  $(\mathbf{z}^\varphi, \mathbf{a}^\varphi) \leftarrow (\mathbf{z}^\varphi, op(\mathbf{a}^\varphi))$ 
16:    end if
17:  update  $D \leftarrow D \cup (\mathbf{z}^\varphi, \mathbf{a}^\varphi)$ 
18:  end for
19:  rederive  $\pi \leftarrow \text{policyDerivation}(D)$ 
20: end while
21: return  $\pi$ 

```

Algorithm 1 presents pseudo-code for the A-OPI algorithm. To begin, an initial policy π is derived from the set of teacher demonstrations (line 2). A single practice run (lines 3-20) consists of a single execution-feedback-update cycle.

During the learner execution portion of a practice run (lines 5-9), the learner executes the task. At each timestep the learner observes the world, and predicts action \mathbf{a}^t according to policy π (line 6). This action is executed and recorded in the *prediction* trace d , along with observation \mathbf{z}^t (line 8). The information recorded in the trace d will be incorporated into the policy update. The global position x^t, y^t and heading θ^t of the mobile robot is additionally recorded, into the *position* trace tr .

Information recorded in tr will be used by the F3MRP framework, when visually depicting the path taken by the robot on the ground during execution.

During the teacher feedback portion of the practice phase, the teacher first indicates, through the F3MRP interface, a segment Φ of the learner execution trajectory requiring improvement. The teacher further indicates an advice-operator op , selected from a finite list, to correct the execution within this segment (line 10).

The teacher feedback is then applied across all points recorded in d and within the indicated subset Φ (lines 11-18). For each point $(\mathbf{z}^\varphi, \mathbf{a}^\varphi) \in d$, $\varphi \in \Phi$, the algorithm modifies either its observation (line 13) or action (line 15), depending on the type of the indicated advice-operator. The modified datapoints are added to the demonstration set D (line 17), and the policy is rederived (line 19).

3 Empirical Validation of A-OPI

This section presents an empirical validation of our corrective feedback approach. We validate the A-OPI algorithm that employs our corrective feedback techniques - that is, advice-operators and the F3MRP framework - on a Segway Robot Mobility Platform (RMP) [28] performing a spatial positioning task. Policy modifications due to corrective feedback are shown to improve policy performance, which furthermore is found to be similar or superior to the more typical approach of providing more teacher demonstrations.

3.1 Experimental Setup

Empirical validation of the A-OPI algorithm is performed through a spatial positioning task with a Segway RMP. This section presents the task and domain, followed by policy development and evaluation; further empirical details may be found in [3].

3.1.1 Task and Domain

The Segway RMP is a two-wheeled dynamically-balancing differential drive robot, which may only drive forward or turn and cannot go sideways. The robot accepts wheel speed commands, but does not allow access to its balancing control mechanisms. We therefore treat Segway RMP control as a black box, since we do not know the specific gains or system parameter values. The inverted pendulum dynamics of the robot present an additional element of uncertainty for low level motion control.

The spatial positioning task consists of attaining a 2-D planar target position (x_g, y_g) with a heading θ_g (Fig. 3). 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 for this specific robot platform the desired motion

trajectories for our task 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.



Fig. 3 Segway RMP performing the spatial positioning task (approximate ground path in yellow).

To gather demonstration examples, a human teleoperates the platform as the robot records from its own sensors. Teleoperation minimizes correspondence issues between demonstrator and learner, and is reasonable to perform for this task and robot platform. The robot observes its global position and heading through wheel encoders sampled at $30Hz$.

To derive a policy from the demonstration examples, the function mapping states to actions is directly approximated via regression techniques. We employ a form of Locally Weighted Learning [7]. Worthwhile to note however is that A-OPI is not restricted to a particular regression technique, and *any* are appropriate for use within the algorithm. Given observation \mathbf{z}^t , action \mathbf{a}^t is predicted through an averaging of the actions in D , weighted by a kernelized distance between their associated datapoint observations and the current observation \mathbf{z}^t . Thus,

$$\mathbf{a}^t = \sum_{(\mathbf{z}_i, \mathbf{a}_i) \in D} \phi(\mathbf{z}^t, \mathbf{z}_i) \cdot \mathbf{a}_i, \quad \phi(\mathbf{z}^t, \mathbf{z}_i) = \frac{e^{(\mathbf{z}_i - \mathbf{z}^t)^T \Sigma^{-1} (\mathbf{z}_i - \mathbf{z}^t)}}{\sum_{\mathbf{z}_j \in D} e^{(\mathbf{z}_j - \mathbf{z}^t)^T \Sigma^{-1} (\mathbf{z}_j - \mathbf{z}^t)}} \quad (1)$$

where the weights $\phi(\mathbf{z}^t, :)$ are normalized over i . In this work the distance computation is Euclidean, the kernel is Gaussian and Σ^{-1} is a constant diagonal matrix that scales each observation dimension and embeds the bandwidth of the Gaussian kernel. All parameters are tuned through Leave-One-Out-Cross-Validation (LOOCV), minimizing the least squared error of the regression prediction on the set D .

The observations for this task are 3-dimensional, and are feature computations involving the global and target position and heading: (i) squared Euclidean distance to the target position, (ii) angle between the target position and current robot heading and (iii) angle between the current and target robot headings. The actions are 2-dimensional: (i) translational speed and (ii) rotational speed. The motion control operators developed for this domain adjust observation inputs (Tbl. 1, Operator 0), single action dimensions by non-static amounts (Operators 1-6) or multiple action

dimensions by non-static amounts (Operators 7-8). The amount of the non-static adjustments are determined as a function of the executed values of the observations and actions.

Table 1 Advice-operators for the spatial positioning task.

	Operator	Parameter
0	Reset goal, recompute observation	
1	No turning	
2	Start turning	[cw ccw]
3	Smooth rotational speed	[dec inc]
4	No translation	
5	Smooth translational speed	[dec inc]
6	Translational [ac/de]celeration	[dec inc]
7	Turn tightness	[less more]
8	Stop all motion	

Key: (c)cw=(counter)clockwise, (dec/inc)=(de/in)crease

3.1.2 Policy Development and Evaluation

The set D is seeded with demonstrations recorded as the teacher teleoperates the robot learner (9 demonstrations, totaling 900 datapoints). Policy improvement proceeds as follows. A random goal is selected (without replacement) from a practice set consisting of (x_g, y_g, θ_g) goals, drawn uniformly within the bounds of the demonstration dataset. The robot executes with its current policy to attain this goal. The advisor observes this execution, and optionally offers policy improvement information. The policy is re-derived, and drawing a new goal initiates another practice run.

Three policies are developed using distinct techniques, differing in *what* is offered as policy improvement information. A total of four policies are therefore developed within this domain:

1. *Baseline Policy (Base)*: Derived from the initial demonstration set.
2. *Feedback Policy (FB)*: Provided with policy corrections, via advice-operators.
3. *Feedback-Hybrid Policy (FB-H)*: Initially provided with more teacher demonstrations; later provided with policy corrections via advice-operators.
4. *More-Demonstration Policy (M-Demo)*: Provided with more teacher demonstrations.

The final three are referred to collectively as the *improvement policies*. Note that in the case of policy *M-Demo*, a practice run consists of a single execute-*demonstrate*-update cycle.

Policies are evaluated for accuracy and success, on an independent test set of (x_g, y_g, θ_g) goals. Here *accuracy* is defined as Euclidean distance between the final

robot and goal positions $e_{x,y}$, and the final robot and goal headings e_θ . *Success* is defined generously as $e_{x,y} < 1.0\text{ m}$ and $e_\theta < \frac{\pi}{2}\text{ rad}$. Practice runs were halted once performance on the test set no longer improved (number of practice runs: 60 *FB*, 59 *FB-H* and 51 *M-Demo*).

3.2 Results

Policy performance was found to improve with corrective feedback, in both execution success and accuracy [3]. A-OPI additionally enabled similar or superior performance when compared to a policy derived from more teacher demonstrations. Furthermore, by concentrating new data exclusively to the areas visited by the robot and needing improvement, A-OPI produced noticeably smaller datasets.

3.2.1 Success and Accuracy Improvement

Table 2 presents the percent execution success of each policy on the independent test set. When compared to policy *Base*, all policy improvement approaches display an increase in success. Both of the feedback policies additionally achieve higher success than policy *M-Demo*.

Table 2 Execution Percent Success

Baseline	Feedback	Feedback-Hybrid	More-Demonstration
32	88	92	80

Figure 4 plots, for each policy, the average position and heading error on the test set goals. For positional error, all improvement policies displayed similar performance, which was a dramatic improvement over policy *Base*. For heading, policy *FB* reduced more error than policy *FB-H*, with both showing marked improvements over policy *Base*. By contrast, policy *M-Demo* displayed *no* improvement in heading error over policy *Base*. That heading error was in general more difficult to improve than positional error is consistent with our prior experience with this robot platform, which is highly sensitive to the accumulation of rotational dead reckoning error.

The iterative nature of policy development under A-OPI produces many intermediate policies, a sampling of which were also evaluated on the test set. Figure 5 shows the average position and heading error of the intermediate policies on the test set goals, to mark the progress of each policy improvement technique.

Superior heading performance was consistently produced by corrective feedback, with policy *FB* attaining lower heading error than policy *M-Demo* throughout policy improvement. By contrast, initially greater improvement in positional error is seen with more demonstration and thus with policy *M-Demo*. While corrective feedback

Test Set Error, Final Policies

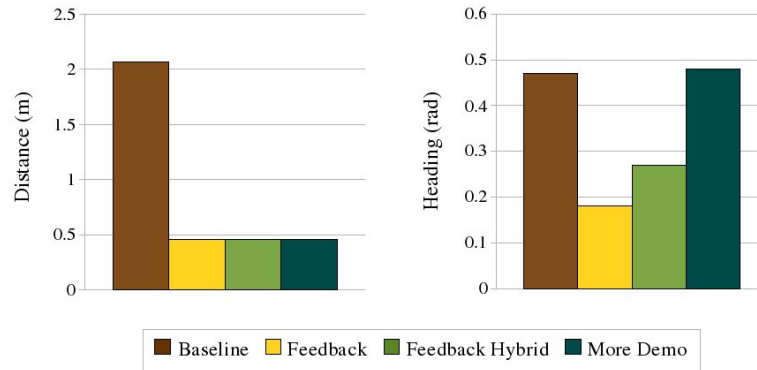


Fig. 4 Average test set error on target position (left) and heading (right), with the *final* policies.

reduces positional error more slowly, policy *FB* does however eventually converge to the level attained through more demonstration.

Policy *FB-H* initially displays the superior reduction in positional error, and inferior reduction in heading error, of policy *M-Demo*. This performance is followed by substantial reductions in heading error, akin to policy *FB*. These results reflect to the development technique of policy *FB-H*. The policy was initially seeded with an intermediate version of policy *M-Demo* (resulting after 23 practice runs), and following the seeding was offered exclusively corrective feedback.

Test Set Error, Intermediate Policies

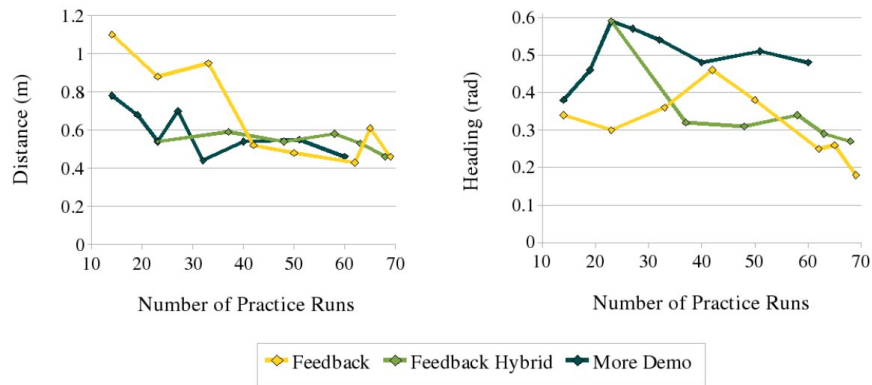


Fig. 5 Average test set error on target position (left) and heading (right), with *intermediate* policies.

3.2.2 More Focused Datasets

How many datapoints were added with each practice run varied greatly depending on whether the execution received corrective feedback or more demonstrations (Fig. 6). The reason is that, in contrast to teleoperation, only subsets of a corrected execution were added to the dataset; in particular, only those execution points which actually received corrections. States visited during good performance portions of the student execution were *not* redundantly added to the dataset. In this manner, the final policy performances shown in Figure 4 were achieved with much *smaller* datasets for both feedback policies, in comparison to policy *M-Demo*. Note that the results of Figure 5 are plotted against the number of *practice runs* contributing to the dataset, and not the number of *datapoints* in the set.

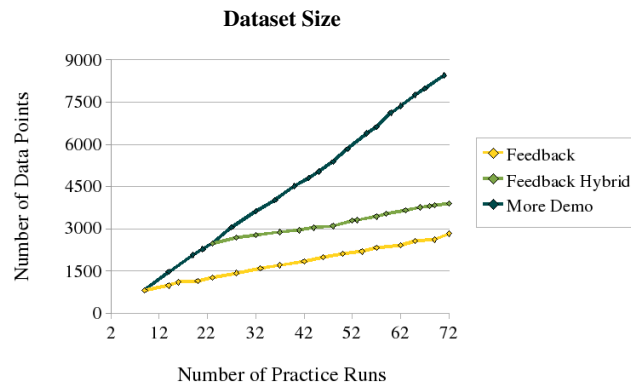


Fig. 6 Growth in dataset size with practice runs.

4 Conclusion

To define an algorithm for motion control on a mobile robot is a difficult and challenging task, which we address by continuing to adapt and refine a control policy based on execution experience. Our approach to motion control demonstrates a behavior to the robot, and addresses potential limitations in the resultant dataset through execution experience. In particular, policy refinement is achieved through corrective feedback provided by a human teacher.

There are two key challenges to providing feedback within low-level motion control domains. The first is the continuity of the action space: continuous-valued actions require continuous-valued corrections, and thus selection from an infinite set. The second is the sampling rate of the policy: a rapid sampling rate means that multiple execution points are responsible for a particular behavior being corrected. We have developed techniques to address each of these challenges.

The first technique, named *advice-operators*, is a language through which a human teacher provides corrections to a robot student. Advice-operators perform mathematical computations on continuous-valued datapoints. To provide a correction, the teacher selects from a finite list of advice-operators. The robot learner applies the operator to an execution datapoint, modifying its value and producing a continuous-valued correction.

The second technique, named *Focused Feedback for Mobile Robot Policies (F3MRP)*, is a framework through which a human teacher provides feedback on mobile robot motion control executions. Through the F3MRP interface, the teacher selects segments of the execution to receive feedback, which simplifies the challenge of providing feedback to policies sampled at a high frequency. A crucial element of the interface is the visual presentation of the ground path taken by the robot during execution; the framework thus targets *mobile* robots in particular.

By pairing these two techniques, the selection of a *single* advice-operator and application segment therefore provides *continuous*-valued corrections on *multiple* execution points. In this manner, our approach enables correction-giving that is reasonable and effective for a human to provide, even within a continuous-valued, rapidly sampled, domain.

We have validated these techniques through our *Advice-Operator Policy Improvement (A-OPI)* algorithm, which employs both advice-operators and the F3MRP framework. A-OPI was implemented on a Segway RMP robot, performing a spatial positioning task. Within this domain, corrective feedback was found to improve policy performance, and to enable similar or superior performance when compared to a policy derived from more teacher demonstrations. Furthermore, by concentrating new data exclusively to the areas visited by the robot and in need of improvement, corrective feedback also produced noticeably smaller datasets, and without a sacrifice in policy performance, suggesting the datasets to be more focused and contain less redundant data.

Acknowledgements The research is partly sponsored by the Boeing Corporation under Grant No. CMU-BA-GTA-1, BBNT Solutions under subcontract No. 950008572, via prime Air Force contract No. SA-8650-06-C-7606, the United States Department of the Interior under Grant No. NBCH-1040007 and the Qatar Foundation for Education, Science and Community Development. The views and conclusions contained in this document are solely those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of any sponsoring institution, the U.S. government or any other entity.

References

1. P. Abbeel, A. Coates, M. Quigley, and A. Y. Ng. An application of reinforcement learning to aerobatic helicopter flight. In *Proceedings of Advances in Neural Information Processing (NIPS '07)*, 2007.
2. P. Abbeel and A. Y. Ng. Exploration and apprenticeship learning in reinforcement learning. In *Proceedings of the 22nd International Conference on Machine Learning (ICML '05)*, 2005.

3. B. Argall, B. Browning, and M. Veloso. Learning robot motion control with demonstration and advice-operators. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '08)*, 2008.
4. B. Argall, S. Chernova, B. Browning, and M. Veloso. A survey of robot learning from demonstration. *Robotics and Autonomous Systems*, 57(5):469–483, 2009.
5. B. Argall, Y. Gu, B. Browning, and M. Veloso. The first segway soccer experience: Towards peer-to-peer human-robot teams. In *First Annual Conference on Human-Robot Interactions (HRI '06)*, 2006.
6. B. D. Argall. *Learning Mobile Robot Motion Control from Demonstration and Corrective Feedback*. PhD thesis, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, 2009. Technical report CMU-RI-TR-09-13.
7. C. G. Atkeson, A. W. Moore, and S. Schaal. Locally weighted learning. *Artificial Intelligence Review*, 11:11–73, 1997.
8. C. G. Atkeson and S. Schaal. Robot learning from demonstration. In *Proceedings of the Fourteenth International Conference on Machine Learning (ICML '97)*, 1997.
9. J. A. Bagnell and J. G. Schneider. Autonomous helicopter control using reinforcement learning policy search methods. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA '01)*, 2001.
10. D. C. Bentivegna. *Learning from Observation Using Primitives*. PhD thesis, College of Computing, Georgia Institute of Technology, Atlanta, GA, 2004.
11. A. Billard, S. Callinon, R. Dillmann, and S. Schaal. Robot programming by demonstration. In B. Siciliano and O. Khatib, editors, *Handbook of Robotics*, chapter 59. Springer, New York, NY, USA, 2008.
12. C. Breazeal and B. Scassellati. Robots that imitate humans. *Trends in Cognitive Sciences*, 6(11):481–487, 2002.
13. B. Browning, L. Xu, and M. Veloso. Skill acquisition and use for a dynamically-balancing soccer robot. In *Proceedings of 19th National Conference on Artificial Intelligence (AAAI '04)*, 2004.
14. S. Calinon and A. Billard. Incremental learning of gestures by imitation in a humanoid robot. In *Proceedings of the 2nd ACM/IEEE International Conference on Human-Robot Interactions (HRI '07)*, 2007.
15. S. Chernova and M. Veloso. Learning equivalent action choices from demonstration. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '08)*, 2008.
16. S. Chernova and M. Veloso. Multi-thresholded approach to demonstration selection for interactive robot learning. In *Proceedings of the 3rd ACM/IEEE International Conference on Human-Robot Interaction (HRI '08)*, 2008.
17. K. Dautenhahn and C. L. Nehaniv, editors. *Imitation in animals and artifacts*. MIT Press, Cambridge, MA, USA, 2002.
18. Y. Demiris and G. Hayes. Imitation as a dual-route process featuring predictive and learning components: a biologically-plausible computational model. In Dautenhahn and Nehaniv [17], chapter 13.
19. D. H. Grollman and O. C. Jenkins. Dogged learning for robots. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA '07)*, 2007.
20. A. J. Ijspeert, J. Nakanishi, and S. Schaal. Learning rhythmic movements by demonstration using nonlinear oscillators. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '02)*, 2002.
21. A. J. 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 (ICRA '02)*, 2002.
22. J. Z. Kolter, P. Abbeel, and A. Y. Ng. Hierarchical apprenticeship learning with application to quadruped locomotion. In *Proceedings of Advances in Neural Information Processing (NIPS '08)*, 2008.
23. D. Kulic and Y. Nakamura. Incremental learning of full body motion primitives. In Peters and Sigaud [31].

24. M. Lopes, F. Melo, L. Montesano, and J. Santos-Victor. Cognitive processes in imitation: Overview and computational approaches. In Peters and Sigaud [31].
25. M. J. Mataric. Sensory-motor primitives as a basis for learning by imitation: Linking perception to action and biology to robotics. In Dautenhahn and Nehaniv [17], chapter 15.
26. C. L. Nehaniv and K. Dautenhahn. The correspondence problem. In Dautenhahn and Nehaniv [17], chapter 2.
27. A. Y. Ng, A. Coates, M. Diel, V. Ganapathi, J. Schulte, B. Tse, E. Berger, and E. Liang. Inverted autonomous helicopter flight via reinforcement learning. In *International Symposium on Experimental Robotics*, 2004.
28. H. G. Nguyen, J. Morrell, K. Mullens, A. Burmeister, S. Miles, K. Thomas, and D. W. Gage. Segway robotic mobility platform. In *SPIE Mobile Robots XVII*, 2004.
29. M. N. Nicolescu and M. J. Mataric. Methods for robot task learning: Demonstrations, generalization and practice. In *Proceedings of the Second International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS '03)*, 2003.
30. J. Peters and S. Schaal. Natural actor-critic. *Neurocomputing*, 71(7-9):1180–1190, 2008.
31. J. Peters and O. Sigaud, editors. *From Motor to Interaction Learning in Robots*. Springer, New York, NY, 2009.
32. N. Ratliff, D. Bradley, J. A. Bagnell, and J. Chestnutt. Boosting structured prediction for imitation learning. *Proceedings of Advances in Neural Information Processing Systems (NIPS '07)*, 2007.
33. P. E. Rybski, K. Yoon, J. Stolarz, and M. M. Veloso. Interactive robot task training through dialog and demonstration. In *Proceedings of the 2nd ACM/IEEE International Conference on Human-Robot Interactions (HRI '07)*, 2007.
34. W. D. Smart. *Making Reinforcement Learning Work on Real Robots*. PhD thesis, Department of Computer Science, Brown University, Providence, RI, 2002.
35. R. Stefani, B. Shahian, C. Savant, and G. Hostetter. *Design of Feedback Control Systems*. Oxford University Press, 2001.
36. M. Stolle and C. G. Atkeson. Knowledge transfer using local features. In *Proceedings of IEEE International Symposium on Approximate Dynamic Programming and Reinforcement Learning (ADPRL '07)*, 2007.
37. R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, Cambridge, MA, London, England, 1998.