Int J Soc Robot
DOI 10.1007/s12369-012-0156-9

Policy Feedback for the Refinement of Learned Motion Control on a Mobile Robot

Brenna D. Argall · Brett Browning · Manuela M. Veloso

Accepted: 13 June 2012

Abstract Motion control is fundamental to mobile robots, and the associated challenge in development can be assisted by the incorporation of execution experience to increase policy robustness. In this work, we present an approach that updates a policy learned from demonstration with human teacher feedback. We contribute *advice-operators* as a feedback form that provides corrections on state-action pairs produced during a learner execution, and *Focused Feedback for Mobile Robot Policies (F3MRP)* as a framework for providing feedback to rapidly-sampled policies. Both are appropriate for mobile robot motion control domains. We present a general feedback algorithm in which multiple types of feedback, including advice-operators, are provided through the F3MRP framework, and shown to improve policies initially derived from a set of behavior examples. A comparison to providing more behavior examples instead of more feedback finds data to be generated in different areas of the state and action spaces, and feedback to be more effective at improving policy performance while producing smaller datasets.

B.D. Argall (✉)
Depts. of Electrical Engineering & Computer Science and Physical Medicine & Rehabilitation, Northwestern University, 2145 Sheridan Road, Evanston, IL 60208, USA
e-mail: brenna.argall@northwestern.edu

B. Browning
The Robotics Institute, Carnegie Mellon University, 5000 Forbes Ave, Pittsburgh, PA 15213, USA
e-mail: brettb@cs.cmu.edu

M.M. Veloso
Computer Science Department, Carnegie Mellon University, 5000 Forbes Ave, Pittsburgh, PA 15213, USA
e-mail: mmv@cs.cmu.edu

Keywords Robot learning · Mobile robots · Demonstration learning · Motion control

1 Introduction

Robust motion control algorithms are fundamental to the successful, autonomous operation of mobile robots. A control *policy*, mapping observed world state to robot action, is one representation for robot motion control. Even with a carefully crafted policy however, a robot often will not behave as the designer expects or intends in all areas of the execution space. Approaches often are unable to scale well with robot and domain complexity, for example by requiring fully defined dynamics models for high degree of freedom robots or interactions with physically-compliant objects. One way to address behavior shortcomings is to update a policy based on execution experience, which can increase policy robustness and overall performance.

The approach taken in this article provides the motion control policy of a mobile robot with human teacher evaluations of execution performance. We focus on policy learning techniques that derive a policy from a set of behavior examples. In particular, *Learning from Demonstration (LfD)* is an approach in which examples of behavior execution by a teacher are provided to a learner, who derives a control policy from the resulting dataset. Our approach augments demonstration learning by providing policy performance evaluations through multiple forms of human *teacher feedback*, the most noteworthy of which directly corrects a state-action mapping predicted by the policy. Feedback is used by the learner to update its policy, enabling further policy development.

The target domain for our techniques is low-level motion control of a mobile robot. Key challenges to providing

« SORO 12369 layout: Large v.1.3.2 file: soro156.tex (Andrius) class: spr-twocol-v1.4 v.2012/05/30 Prn:2012/07/02; 9:40 p. 2/13»
« doctopic: OriginalPaper numbering style: ContentOnly reference style: basic»

feedback within such a domain are the *rapid sampling rate* and *continuous state-action space* of the policy. With a rapid sampling rate, any exhibited robot behavior deemed by the teacher to need feedback likely executed over multiple (tens or hundreds of) timesteps. With a continuous state-action space, corrective feedback that indicates a preferred action or state requires providing a continuous-valued correction, and thus selection from an infinite set.

In this article, we contribute techniques to address both of these challenges. In particular, we introduce *Focused Feedback For Mobile Robot Policies (F3MRP)* as a framework for providing feedback on a sequence of state-action pairs generated by the execution of a policy sampled at a rapid rate. We contribute *advice-operators* as a mechanism for mapping a finite set containing high-level human feedback to an infinite set of continuous-valued corrections, along with a structured approach for defining the advice-operators.

We provide a general feedback algorithm that makes use of both F3MRP and advice-operators. The algorithm derives an initial policy from a set of behavior examples, within an LfD paradigm. Our work targets addressing some of the limitations potentially present within an LfD dataset; in particular, those which are not practical or possible to address through more demonstration alone. Overviews of three variants of our general feedback algorithm are provided, along with a summary of their empirical validations. We present in detail an empirical comparison between data produced under our approach to data produced exclusively from teacher demonstration. The two techniques are shown to produce data in different areas of the state and action spaces. We furthermore establish the combined techniques of advice-operators and F3MRP to be an effective and efficient approach for policy improvement.

We begin in Sect. 2 with a presentation of the related literature that supports this work. Section 3 highlights and discusses key features that characterize a feedback framework, and introduces both advice-operators and F3MRP. The general feedback algorithm that employs both of these techniques is presented in Sect. 4, along with an overview of multiple implementation variants. In Sect. 5 data produced under our approach is compared to data produced using demonstration alone. Our approach to advice-operator development is described in Sect. 6, and we conclude the article by highlighting future research directions.

2 Background and Motivation

We frame the motivation for corrective feedback within the context of demonstration learning. Note however that the majority of the dataset limitations listed in the following section are potentially present in *any* set of behavior examples, and not only those produced from demonstration. Moreover, the advice-operator correction technique is appropriate for the refinement of policies produced from any behavior example set, and its applicability thus extends beyond the domain of LfD.

We consider LfD [5, 10] formulations that have a teacher demonstrate a desired behavior to the robot to produce a dataset of sequences of observation-action pairs, from which a control policy is derived. The policy $\pi : Z \rightarrow A$ maps world observations $Z \in \mathbb{R}^n$, computed from sensor readings, to robot actions $A \in \mathbb{R}^m$. Real world uncertainty means that multiple demonstrations of the same behavior will not execute identically, and so generalization over the examples produces a policy that does not depend on a strictly deterministic world, and thus is more robust to this uncertainty. Demonstration has many additional attractive features for both learner and teacher, including a relaxation on the requirements of robotics expertise and explicit model (world or robot) specifications, as well as focusing the dataset of examples to areas of the state-action space actually encountered during behavior execution.

Techniques for recording a demonstration dataset cover a broad range, from the teacher directly controlling the passive robot platform while recording from its sensors [8], to having the teacher's body execute the task while recorded by sensors external to [9] or located on the body of [16] the teacher. Following demonstration, a policy is derived from the recorded dataset, via a variety of approaches: for example, to directly approximate the function mapping observations to actions [13, 15, 16, 19], or to learn a cost function for use with a navigational planner or *Reinforcement Learning* (*RL*) paradigms [1, 7, 18, 24]. The function approximation approach is employed in our work, using regression techniques to predict within our continuous action space.

Though LfD has enabled successful policy development for a variety of robotics applications, the approach is not without its limitations, which can include:

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

Dataset sparsity is the trade-off to focusing the dataset to state-space areas visited during task execution, and might be partly overcome by the generalization ability of the policy derivation technique. In the event of poor quality dataset examples a variety of sources might be responsible, including the demonstration abilities of the teacher or limitations in the interface used for control during demonstration. Since demonstration for real robots involves executing actions in physical environments, differences in embodiment between the learner and teacher become of crucial importance, and the challenges that arise as a result, when mapping teacher

Journal ID: 12369, Article ID: 156, Date: 2012-07-02, Proof No: 3, UNCORRECTED PROOF

« SORO 12369 layout: Large v.1.3.2 file: soro156.tex (Andrius) class: spr-twocol-v1.4 v.2012/05/30 Prn:2012/07/02; 9:40 p. 3/13»
« doctopic: OriginalPaper numbering style: ContentOnly reference style: basic»

demonstrations to the learner, are broadly referred to as *correspondence issues* within the literature [11, 20].

One successful approach for dealing with poor or ambiguous teacher demonstrations is to provide more demonstration data in response to execution experience with the policy [12, 13, 15]. Another approach is to pair LfD with Reinforcement Learning techniques, for example by restricting the RL search space to states that lie near the demonstrations [17, 22] or employing an exploration policy to visit undemonstrated states [23, 25]. A final approach is to directly correct the behavior exhibited by the policy [13, 21]. Note that these cited LfD correction paradigms operate within action spaces that are *discrete* and with actions of significant time duration, sampled with *low* frequency. By contrast, our work falls into the notably challenging category of correcting policies within *continuous* action-spaces sampled at *high* frequency, which is largely unaddressed within the LfD literature.

3 Corrections for Rapidly Sampled Policies

This section begins by introducing *advice-operators* as a mechanism for translating high-level corrections into low-level continuous-valued execution modifications (Sect. 3.1). With advice-operators, we aim to address the challenge of providing policy corrections within continuous state-action spaces. *Focused Feedback For Mobile Robot Policies (F3MRP)*[1] is afterwards introduced as a framework through which portions of a policy execution are selected to receive feedback (Sect. 3.4). With F3MRP, we aim to address the challenge of providing feedback to policies sampled at a rapid rate.

We furthermore identify and discuss the following as key characterizations of a feedback framework: the *type* of the feedback (Sect. 3.2) and the *interface* for providing feedback (Sect. 3.3). According to which, our feedback approach may be characterized as the following:

Feedback type: The types of feedback include policy correction via advice-operators and binary performance flags. Most commonly, the feedback produces a new state-action example mapping, which is added to the policy dataset prior to rederivation.

Feedback interface: Evaluations are performed by a human teacher, who selects a piece of feedback and path segments from a graphical display of the robot execution path. F3MRP associates these segments with portions of the observation-action sequence, which are passed with the feedback to the learner.

[1]The F3MRP framework was developed within the GNU Octave scientific language [14].

3.1 Advice-Operators

Explicit corrections can be a very direct approach to policy improvement, by indicating a preferred action to take (or state to enter) and therefore not requiring any exploration on the part of the learner, unlike state reward. Expecting a human to know the appropriate continuous *value* that corrects an execution point however is neither reasonable nor efficient. We therefore contribute *advice-operators* [2] as a language through which a human teacher provides high-level policy corrections on continuous-valued policy behaviors.

Concretely defined, an advice-operator is a mathematical computation performed on an observation input or action output. An operator $f : (\mathbf{z}, \mathbf{a}) \rightarrow (\hat{\mathbf{z}}, \hat{\mathbf{a}}) \in \mathbb{R}^{n+m}$ maps an observation-action pair within the joint space of observations and actions. The set $F = \{f_i\}_{i=1}^{N_p}$ of N_p operators are defined commonly between the learner and advisor. In our implementations, the operator mapping f is defined via a structured approach that will be presented in Sect. 6, along with an example set of advice-operators used in our empirical work. The teacher furthermore must indicate a segment of the learner execution, accomplished in our approach via the F3MRP framework, introduced next (Sect. 3.4). To illustrate with a simple example, consider an operator f that modifies each action dimension by a multiplicative amount $\alpha \in \mathbb{R}$, applied over a segment of 50 datapoints; then, $f(\mathbf{z}^t, \mathbf{a}^t) = (\mathbf{z}^t, \alpha \cdot \mathbf{a}^t), \forall t = 1..50$.

Note that the actual control policy formulation is unknown to the human advisor, save for its expression via the observation-action pairs seen during a learner execution. Corrections are therefore offered on these executed observation-action pairings. A key insight to the advice-operator approach is that pairing a modified observation (or action) with the *executed* action (or observation) now represents a corrected mapping, and thus a correction for the policy itself. How to best incorporate the corrected mapping into the policy is an open topic. We take the straightforward approach of adding a corrected datapoint to the demonstration set and rederiving, which steps the function approximation in a direction that corrects the policy as well.[2]

3.2 Feedback Type

We now consider the topic of feedback on policy performance more generally. Feedback in our work consists primarily of advice-operators, but in more general terms feedback can take a variety of forms. For example, the learner

[2]The empirical validations of Sect. 4.2 employ lazy learning regression techniques [6]; specifically, a form of locally weighted averaging. Incremental policy updating is particularly straightforward under lazy learning regression, since explicit rederivation is not required; policy derivation happens at execution time and so a complete policy update is accomplished by simply adding new data to the set.

Journal ID: 12369, Article ID: 156, Date: 2012-07-02, Proof No: 3, UNCORRECTED PROOF

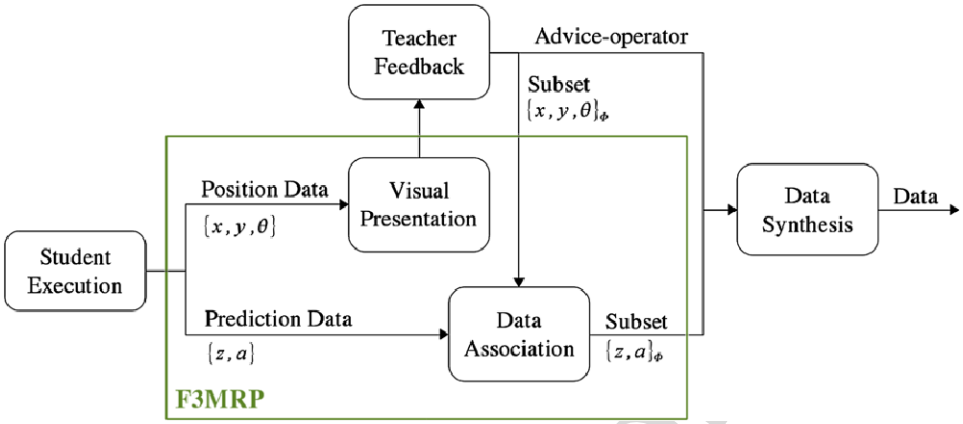

Fig. 1 Path visualization, subset selection and data association in the F3MRP framework

might receive a single reward upon reaching a failure state, or the value of a gradient along which a more desirable action may be found. The feedback type determines *what* is provided to the learner, which we propose to characterize according to three axes:

1. The **amount of information** the feedback encodes. Feedback can encode some, none or all of the translation from policy evaluation to policy update.
2. The **continuity** of the feedback. The value taken by feedback might be discrete (possibly binary) and derive from a finite set, or continuous and derive from an infinite set.
3. The **frequency** at which feedback is provided. Governed by whether feedback is provided for entire executions, subsets, or individual decision points, and the corresponding time durations of each.

For succinctness, we refer to the combination of continuity and frequency as the *granularity* of the feedback.

The *amount of information* contained within feedback spans a continuum, where at one extreme feedback provides very minimal information and the majority of the work of policy refinement lies with the learner. This is the case in the single failure-state reward example, where to translate the feedback into a policy update the learner must employ techniques like RL to incorporate the reward, and furthermore must determine through exploration which alternate states improve policy performance. At the opposite extreme feedback provides very detailed information, where the majority of the work of policy refinement is encoded within the feedback details. This is the case with the action-gradient example, where to determine an improved policy prediction the learner needs only to adjust the function approximation of its policy along this gradient. In practice, policy execution consists of multiple phases, beginning with sensor readings being processed into state observations and ending with the execution of a predicted action. The *granularity* of the phase receiving feedback drives the granularity of the feedback itself. For example, consider a policy sampled at low frequency (e.g. tens of seconds or minutes), with binary observations that monitor the presence of specific objects and feedback that draws attention to these objects (e.g. [21]). In this case the teacher provides feedback on a *single* decision point, since the policy is sampled infrequently, and by selecting from a *discrete* set, i.e. those objects being monitored by the policy.

To place the advice-operator feedback type within this characterization, the *amount of information* is large since an explicit indication of the preferred state-action mapping is provided. The feedback *granularity* is fine, since continuous-valued corrections are provided at high frequency (in our work 30 Hz). Other feedback types, of coarser *continuity*, implemented within the F3MRP framework include binary performance flags [3, 4]; where positive or negative credit provides a *binary* indication of whether policy performance in a particular area of the state-space is desirable or not.[3]

3.3 Feedback Interface

The feedback framework serves as an interface between the policy execution, the policy evaluator and the learner. For example, the interface might present the execution in a meaningful format for the policy evaluator, or translate the feedback into meaningful information for the learner. The feedback interface controls *how* feedback is provided. We propose the following characterizations for a *feedback interface*:

1. How the execution is **evaluated**. Governed by the evaluation *source* (e.g. automated computation or task expert) and the *information* the source requires.
2. How closely feedback is **associated** with the execution data. Influenced by the policy sampling rate and to what extent feedback is offered in real-time.

[3]The positive credit flag adds the execution point, unmodified, to the dataset; and thus may equivalently be viewed as an identity function advice-operator, i.e. $f(\mathbf{z}, \mathbf{a}) = (\mathbf{z}, \mathbf{a})$.

« SORO 12369 layout: Large v.1.3.2 file: soro156.tex (Andrius) class: spr-twocol-v1.4 v.2012/05/30 Prn:2012/07/02; 9:40 p. 5/13»
« doctopic: OriginalPaper numbering style: ContentOnly reference style: basic»

3. How feedback is **transferred** to the learner.

The first role of a feedback interface is to assist in the *evaluation* of a given policy execution, and different approaches vary in the amount and type of information they require. For example, a reward computation might require performance statistics like position accuracy. The second role of the feedback interface is to *associate* the evaluation with the underlying execution, which depends on how closely an evaluation is tied to the execution data. For example, an overall performance measure associates with the execution as a whole, and thus links to the data at a very coarse scale.[4] Close feedback-data associations are necessarily influenced by the sampling rate of the policy, and actions with significant time durations are easier to isolate as responsible for a particular policy behavior than actions sampled at a high frequency. The third role of the interface is to *transfer* the execution-associated feedback to the learner. At one extreme, streaming feedback is provided as the learner executes and immediately updates the policy. At the opposite extreme, feedback is provided post-execution, possibly after multiple executions, and updates the policy offline.

3.4 Focused Feedback for Mobile Robot Policies

We now introduce our framework for providing feedback to build and improve motion control policies on a mobile robot, named *Focused Feedback for Mobile Robot Policies (F3MRP)*. An overview of its operation, in the specific case of advice-operator feedback, is provided in Fig. 1. During a student execution, data relating to the robot position $\{x, y, \theta\}$ and policy prediction $\{\mathbf{z}, \mathbf{a}\}$ is recorded. The position data is used by F3MRP for a visual presentation of a robot path, which assists the teacher in the selection of the path subset $\{x, y, \theta\}_\Phi$ requiring correction. This path subset is then associated with the corresponding prediction data subset $\{\mathbf{z}, \mathbf{a}\}_\Phi$, which is provided as input to the selected advice-operator, with the result of synthesized data.

Execution *evaluation* is accomplished by a human teacher observing the learner policy execution from two viewpoints: her own visual observation, and a graphical replay of the 2-D ground path taken by the mobile robot. The visual path presentation is a key component in the identification of those portions of the learner execution that are to receive feedback. The presentation employs a color scheme to visually indicate the *dataset support* of the policy predictions; that is, of how well the dataset D, from which the policy derives,

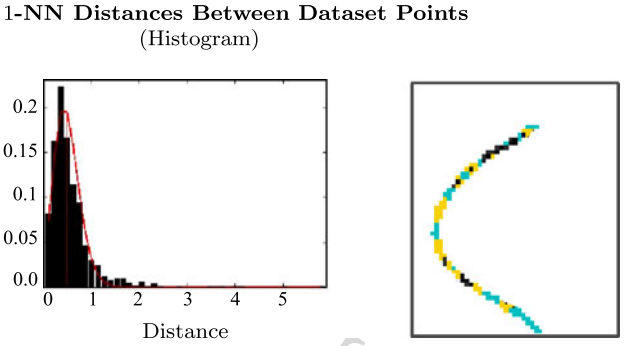

Fig. 2 *Left*: Example distribution of 1-NN distances within a demonstration dataset (*black bars*), and the Poisson model approximation (*red curve*). *Right*: Example plot of the F3MRP display of 2-D ground path, with color indications of dataset support. Given an execution query point $\mathbf{z}^t$ with distance ℓ^t to the demonstration set, plotted in *yellow* are the points for which $\ell^t < \tau_{\kappa=1}$, in *blue* those within $\tau_{\kappa=1} \leq \ell^t < \tau_{\kappa=3}$, and in *black* those for which $\tau_{\kappa=3} \leq \ell^t$ (Color figure online)

covers the observation space in the area of query points observed during the execution. In particular, the distance between query point $\mathbf{z}^t$ and the single *nearest* (Euclidean distance) dataset point contributing to the policy's regression prediction is calculated; i.e. $\ell^t = \min_j \|\mathbf{z}^t - \mathbf{z}^j\|, \mathbf{z}^j \in D$. The color scheme then is set based on the relation between this distance ℓ^t and the *support thresholds* τ_κ of the dataset, calculated as follows. For a given dataset with N points, the set of nearest Euclidean distances $\{\ell^i\}_{i=1}^N$ between points in the set – i.e. $\forall \mathbf{z}^i \in D, \ell^i = \min_{j \neq i} \|\mathbf{z}^i - \mathbf{z}^j\|, \mathbf{z}^j \in D$ – are modeled as a Poisson[5] distribution, with mean $\mu = \lambda$ and variance $\sigma^2 = \lambda$. Data support thresholds τ_κ are defined as $\tau_\kappa = \mu + \kappa\sigma$, where κ is set by hand based on empirical evidence. In Fig. 2 for example, two thresholds are defined: $\kappa = 1$ and $\kappa = 3$. The data support information is used by the teacher in his selection of execution segments and feedback types.

In the F3MRP framework the *association* between feedback and the underlying learner execution is tight. The teacher selects problem segments of the graphically displayed ground path, where segment sizes 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 *observation-action trace*, i.e. the sequence of policy input-output pairs (both of which have been recorded during the learner execution). The high frequency at which a motion control policy is sampled however complicates the above requirement of a tight association between teacher feedback and execution data. To address this complication

[4] This scale becomes finer, and association with the underlying data trickier, if a single value is intended to be somehow distributed across only a portion of the execution states; akin to the RL issue of reward back-propagation.

[5] A Poisson formulation was chosen since the distance calculations never fall below, and often cluster near, zero. To estimate λ, frequency counts were computed for k bins (uniformly sized) of distance data ($k = 50$).

Journal ID: 12369, Article ID: 156, Date: 2012-07-02, Proof No: 3, UNCORRECTED PROOF

Algorithm 1 Baseline Feedback Algorithm

1: Given D
2: *initialize* $\pi \leftarrow$ `policyDerivation`(D)
3: **while** *practicing* **do**
4: *initialize* $\xi_d \leftarrow \{\}$, $\xi_p \leftarrow \{\}$
5: **repeat**
6: *predict* $\{ \mathbf{a}^t, \tau^t \} \leftarrow \pi(\mathbf{z}^t)$
7: *execute* $\mathbf{a}^t$
8: *record* $\xi_d \leftarrow \xi_d \cup (\mathbf{z}^t, \mathbf{a}^t)$, $\xi_p \leftarrow \xi_p \cup (x^t, y^t, \theta^t, \tau^t)$
9: **until** done
10: *advise* $\{ F, \hat{\xi}_p \} \leftarrow$ `teacherFeedback(` `F3MRP(`ξ_p`)` `)`
11: *associate* $\hat{\xi}_d \leftarrow$ `F3MRP(` ξ_d, $\hat{\xi}_p$ `)`
12: *update* $D \leftarrow$ `applyFeedback(` D, F, $\hat{\xi}_d$ `)`
13: *rederive* $\pi \leftarrow$ `policyDerivation(` D `)`
14: **end while**
15: **return** π

F3MRP offers an interactive tagging mechanism, that allows the teacher to mark execution points as they display in real-time on the graphical replay of the 2-D path. The tagging mechanism thus enables more accurate syncing between the feedback and the points to which it applies.

4 Our General Feedback Algorithm

We now introduce a general feedback algorithm that makes use of our policy improvement techniques; that is, of providing feedback of various forms, including advice-operators, through the F3MRP framework. We first detail the general algorithm, and then summarize empirical results from three of its variants.

4.1 Algorithm Overview

Our algorithm has a teacher provide feedback on multiple learner executions. A single practice run consists of a single *execution-feedback-update* cycle; that is, of learner execution followed by teacher feedback and policy update. A subsequent practice run is then initiated, during which the learner executes with the new, updated policy. Practice runs continue until the teacher is satisfied with the performance.

Pseudo-code is provided in Algorithm 1. To begin, an initial policy π is derived from the set D of behavior examples, consisting of observation-action pairs (line 2). The source of the dataset not restricted by the algorithm; in our work, the set is produced from teacher demonstration. The regression technique employed for policy derivation likewise is not restricted by the algorithm, and any black box regression algorithm is suitable for use with this approach. In our empirical validations, a form of locally weighted averaging is employed, since incremental policy updating is particularly straightforward under lazy learning regression.

During the learner *execution* portion of a practice run (lines 5–9), the learner executes the task and information is recorded into traces ξ_d and ξ_p. At each timestep the learner observes the world, and predicts action $\mathbf{a}^t \in A$ according to policy π and with data support τ^t ($\equiv \ell^t$ of Sect. 3.4). This action is executed and recorded, along with observation $\mathbf{z}^t \in Z$, into the prediction trace $\xi_d \in \mathbb{R}^{m+n}$. The information recorded in the trace ξ_d will be used for the policy update. The global ground position x^t, y^t and heading θ^t of the mobile robot are additionally recorded, along with support τ^t, into the position trace $\xi_p \in \mathbb{R}^4$. Information recorded into ξ_p will be used by the F3MRP framework, when visually presenting the path taken by the robot on the ground during the execution.[6]

During the *feedback* portion (lines 10–11), the teacher uses the visual presentation of ξ_p provided by the F3MRP interface to indicate a segment $\hat{\xi}_p \subseteq \xi_p$ of the learner execution, along with feedback F for that segment. The interface associates the position segment $\hat{\xi}_p$ with the appropriate segment $\hat{\xi}_d$ of the prediction trace.

During the policy *update* portion (line 12–13), the learner applies feedback F to the segment $\hat{\xi}_d$, and adds the resulting data to D. Rederiving the policy π from this set completes the policy update. The exact details of how the feedback F and prediction segment $\hat{\xi}_d$ are used to update the dataset are particular to each feedback type. In the case of an advice-operator f, each point $(\mathbf{z}^i, \mathbf{a}^i) \in \hat{\xi}_d$ is mapped to a new observation-action pair, $f : (\mathbf{z}^i, \mathbf{a}^i) \rightarrow (\hat{\mathbf{z}}^i, \hat{\mathbf{a}}^i)$, which then is added to the dataset, $D \leftarrow D \cup (\hat{\mathbf{z}}^i, \hat{\mathbf{a}}^i)$.

4.2 Prior Empirical Validation

Several variants on our general feedback algorithm have been empirically validated, a summary of which is provided in this section. We highlight in particular the versatility of teacher feedback, which not only refines demonstrated policies, but also enables more complex policies to be built from simpler behaviors.

Advice-operators were first introduced with the *Advice-Operator Policy Improvement (A-OPI)* algorithm, which refined policies learned from demonstration [2]. Empirical validation[7] was performed on a Segway RMP robot performing planar motion tasks. An initial case study showed improvements in path-following precision and efficiency, and both beyond the abilities of demonstrator, as well as

[6] The traces ξ_d and ξ_p correspond respectively to the "Prediction Data" and "Position Data" in Fig. 1. Similarly, the trace subsets $\hat{\xi}_d = \{x, y, \theta\}_\Phi$ and $\hat{\xi}_p = \{\mathbf{z}, \mathbf{a}\}_\Phi$.

[7] Here an earlier version of F3MRP was employed, that did not provide visual dataset support or interactive tagging.

« SORO 12369 layout: Large v.1.3.2 file: soro156.tex (Andrius) class: spr-twocol-v1.4 v.2012/05/30 Prn:2012/07/02; 9:40 p. 7/13»
« doctopic: OriginalPaper numbering style: ContentOnly reference style: basic»

the emergence of novel behavior characteristics absent from the demonstration set. Full empirical validation with a more complex set of advice-operators furthermore showed similar or superior performance while producing noticeably smaller datasets, when compared to policies developed from exclusively more teleoperation demonstrations.

The *Feedback for Policy Scaffolding (FPS)* algorithm employed teacher feedback to build a complex policy behavior from simpler demonstrated behaviors [4]. Empirical validation was performed within a simulated motion control domain that tasked a differential drive robot to reactively drive along a racetrack. Primitive behavior policies, which represented simple motion components of this task (*turn-left, turn-right, go-straight*), were learned through demonstration and refined via feedback. A policy able accomplish the more complex behavior (to drive the full track) was successfully developed *only after* receiving teacher feedback that served to guide behavior when transitioning between primitive policies. Empirical results showed performance to improve with teacher feedback, and all FPS policies outperformed the comparative policies developed from teacher demonstration alone: in fact, the exclusively demonstrated policies were never able to successfully perform the more complex task behavior.

The *Demonstration Weight Learning (DWL)* algorithm considered different feedback *types* to be distinct data *sources*, selected through an expert learning inspired paradigm [3]. Source weights were automatically determined and dynamically updated by the algorithm, based on the performance of each expert. Empirical validation within the simulated robot racetrack driving domain confirmed data sources to be unequal in their respective performance abilities on this task, and the weighting was found to improve policy performance. Furthermore, source weights learned under the DWL paradigm were consistent with the respective performance abilities of each individual expert.

5 Comparison to More Mapping Examples

We now compare a dataset built using our feedback techniques to one built from demonstration exclusively. Both datasets are seeded initially with demonstration data from teacher executions, and an initial policy is derived. As the learner executes, the teacher[8] observes the performance and offers either: corrective feedback, in the case of the *feedback* dataset, or more demonstrations, in the case of the *more-demonstration* dataset.

The domain of our empirical comparison consists of a simulated differential drive robot within a racetrack environment.[9] The robot is tasked with driving along the racetrack, with two failure criteria: the robot either stops moving or crosses a track boundary. The robot is controlled by setting target translational and rotational speeds. Demonstrations are performed via human teleoperation that decreases or increases the translational or rotational speed (v, ω) of the robot motion. During execution, the robot computes at each timestep a local track representation by fitting a third order polynomial to track border points visually observed in the current and recent-past timesteps. Observation features and action dimensions are detailed in Table 1.

Feedback provided by the teacher includes advice-operators as well as positive credit. The motion control advice-operators for this task are developed using the operator scaffolding approach of Sect. 6, and are presented in Table 2. Data added to the more-demonstration set is produced in the same manner as the initial dataset (namely, teleoperation), except that demonstrations are provided *in response to* learner executions with its current policy.

Table 1 Description of observation and action dimensions for the racetrack driving task

Dim	Observation	Description
0	v^t	current translational speed
1	ω^t	current rotational speed
2	a	Polynomial coefficients of the track border approximation (i.e. $y = ax^3 + bx^2 + cx + d$)
3	b	
4	c	
5	d	

Dim	Action	Description
0	$\hat{v}^t$	predicted translational speed
1	$\hat{\omega}^t$	predicted rotational speed

Table 2 Advice-operators for the racetrack driving task

	Operator	Parameter
0	Modify ω, *Static* (f_δ)	[− +]
1	Modify v, *Static* (f_δ)	[− +]
2	Modify ω, *Fractional* (f_α)	[− +]
3	Modify v, *Fractional* (f_α)	[− +]
4	Modify ω, *Incr. Frac.* (f_β)	[− +]
5	Modify v, *Incr. Frac.* (f_β)	[− +]
6	Adjust ω and v, *Fractional*	[− +]
7	Adjust Turn, *Fractional*	[loosen tighten]
8	Adjust Turn, *Incr. Frac.*	[loosen tighten]

[8] The same teacher (one of the authors) was used to provide both demonstration and feedback.

[9] Full domain, and algorithm, details may be found in [4].

Journal ID: 12369, Article ID: 156, Date: 2012-07-02, Proof No: 3, UNCORRECTED PROOF

« SORO 12369 layout: Large v.1.3.2 file: soro156.tex (Andrius) class: spr-twocol-v1.4 v.2012/05/30 Prn:2012/07/02; 9:40 p. 8/13»
« doctopic: OriginalPaper numbering style: ContentOnly reference style: basic»

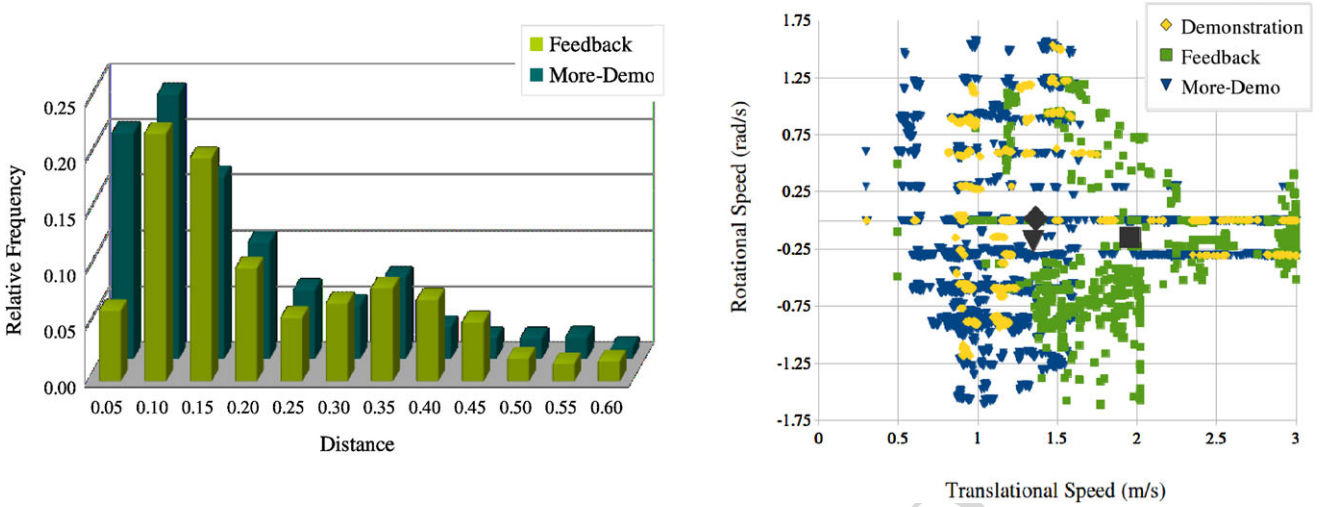

Fig. 3 *Left*: Histogram of observation-space distances between new datapoints and the nearest point within the existing set. *Right*: Plot of each dataset within the action space, with mean values marked in *black*

5.1 Population of the Dataset

To begin, we compare the distribution of data within the observation and action spaces.

Figure 3 (left) displays the *frequency count* of 1-Nearest Neighbor (1-NN) distances within the datasets of each approach. For every new datapoint (1,239 for feedback, 2,520 for more-demonstration), the 1-NN distance was computed as the minimum over the Euclidean distances between its position and that of each point in the existing dataset, within the *observation* space ($\mathbb{R}^6$); i.e. $\forall \mathbf{z}^t, \ell^t = \min_j \|\mathbf{z}^t - \mathbf{z}^j\|, \mathbf{z}^j \in D$. The most notable difference between the two approaches is seen within the frequency of small 1-NN distances. The more-demonstration approach (blue bar) more frequently produced new data that was close (≤ 0.05) to the policy dataset. Furthermore, when considering the *distribution* of 1-NN distances, the more-teleoperation distribution has a lower mean and larger standard deviation (0.15 ± 0.14) than the feedback distribution (0.21 ± 0.02).

These observation space results suggest that our feedback techniques take larger steps away from the initial demonstration set, into more remote areas of the observation space. It is possible that the more-demonstration approach also visits these areas, but over more, smaller-stepped, iterations. Given the performance results presented in the following section however, the more likely explanation is that there exist observation areas which are unvisited by the teacher during demonstration, but which are visited by the learner executions and then added post-correction to the feedback dataset.

Figure 3 (right) provides a visual representation of the distribution of new data within the *action* space ($\mathbb{R}^2$). This figure plots along each axis a single action dimension. Qualitative visual inspection reveals a similar trend to that seen in the observation-space results: namely, that the actions produced through the more-demonstration technique (blue triangles) are closer to the actions already present within the dataset (yellow diamonds) than those actions produced through feedback techniques (green squares). Of particular interest to note is that our feedback techniques produce data within areas of the action space that are entirely *absent* from either demonstration dataset (e.g. around ($1.75 \frac{\text{m}}{\text{s}}, -0.75 \frac{\text{rad}}{\text{s}}$), Fig. 3, right). Possible explanations include that the teacher never encounters situations (i.e. visits areas of the observation space) in which these action combinations are appropriate, or the teacher is unable, or unwilling, to demonstrate them. The feedback dataset does indeed tend to predict faster translational and rotational speeds (Table 3, Action rows), with the result of higher executed speeds (Observation rows 0,1).

Table 3 Mean and variance for each action and observation dimension within the baseline, feedback and more-demonstration datasets. Absolute values computed for rotational speeds (Action dim. ω, Obs. dim. 1)

	Demo	Feedback	More-Demo
Action			
ν ($\frac{\text{m}}{\text{s}}$)	1.4 ± 0.4	2.0 ± 0.4	1.4 ± 0.6
ω ($\frac{\text{rad}}{\text{s}}$)	0.5 ± 0.2	0.5 ± 0.2	0.4 ± 0.2
Observation			
0	1.4 ± 0.4	1.8 ± 0.4	1.3 ± 0.6
1	0.5 ± 0.2	0.5 ± 0.2	0.4 ± 0.2
2	0.0 ± 0.2	-0.1 ± 0.1	-0.1 ± 0.2
3	0.1 ± 0.9	0.1 ± 0.2	0.2 ± 0.5
4	-0.0 ± 0.5	0.0 ± 0.1	-0.1 ± 0.2
5	-0.1 ± 0.1	0.0 ± 0.1	0.0 ± 0.1

Journal ID: 12369, Article ID: 156, Date: 2012-07-02, Proof No: 3, UNCORRECTED PROOF

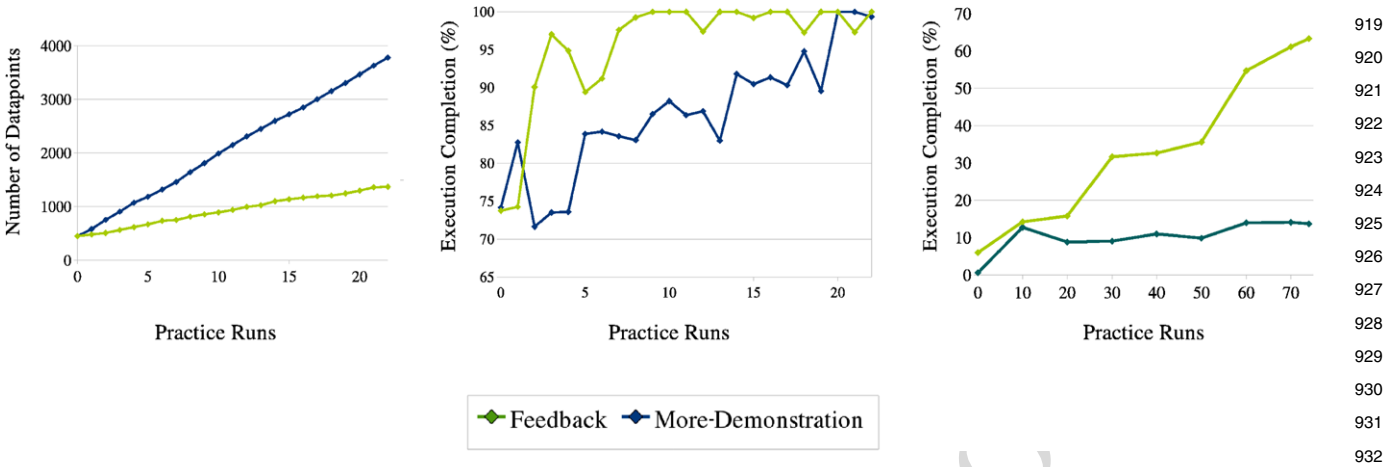

Fig. 4 For the feedback and more-demonstration techniques, across practice runs: the size of each dataset (*left*) and performance improvement of the policies on primitive behaviors (*center*) and a more complex task (*right*)

5.2 Dataset Quality

We next compare the quality of the resultant datasets, as measured by dataset size and policy performance.

Figure 4 (left) presents the growth of each dataset over practice runs. We define a *practice run* as a single execution by the learner, which then receives from the teacher either feedback or an additional demonstration. Note that in the same number of practice runs, the more-demonstration approach (blue line) produces approximately three times the number of datapoints than the feedback approach (green line). For a given learner execution, usually only a portion is responsible for any suboptimal behavior. Our feedback technique corrects only that portion of the execution; by contrast the more-demonstration technique necessarily provides the learner with a complete new execution. For this reason, the more-demonstration technique nearly always[10] adds more points to the dataset.

Figure 4 (center) presents the performance improvement during practice, measured as the percentage of the task successfully completed. Performance of the feedback policies is found to improve more quickly over practice runs, and to be generally superior to those of the more-demonstration policies. Moreover, differences in performance are dramatically pronounced during the development of policies that operate in more complex task domains. Figure 4 (right) presents the policy performance improvement when multiple behavior primitives are scaffolded to perform a more complex task (via the FPS algorithm, Sect. 4.2). The performance of both approaches on the complex task is initially quite poor (0.6 ± 0.2 % more-demonstration, 6.0 ± 0.2 % feedback, average of 10 executions). Across practice runs, the performance of the more-demonstration dataset does improve, marginally, to a 13.7 ± 0.02 % success rate (average of 50 executions). The performance of the feedback dataset is a marked improvement over this, improving to a 63.3 ± 0.3 % success rate (average of 50 executions).

With these performance results, we may conclude that the smaller datasets of our feedback techniques omit primarily redundant data, that do not improve policy performance. Since the feedback policies in fact *exceeded* the more-demonstration policies in performance, we may further conclude that relevant data were *missed* by demonstration, in spite of the larger datasets.

6 Advice-Operator Development

Before concluding, we present our structured approach to defining a set of action advice-operators. The approach first defines a set of baseline operators, and then builds new advice-operators through the scaffolding of existing operators.[11] The synthesized data that these operators produce furthermore is constrained to be firmly grounded within the robot's capabilities. While operators can be task-specific, the baseline (and often also the scaffolded) operators will transfer between task domains, providing the actions themselves do. The baseline operators are as general as the actions they modify.

6.1 Baseline Advice-Operators

Advice-operator development begins with the definition of a baseline set of operators. For multi-dimensional ac-

[10] The exceptions being when the entire learner execution receives a correction, or when the teacher provides a demonstration for only the beginning portion of an execution.

[11] In Table 2, operators 0–5 are the baseline operators and operators 6–8 were built through operator-scaffolding.

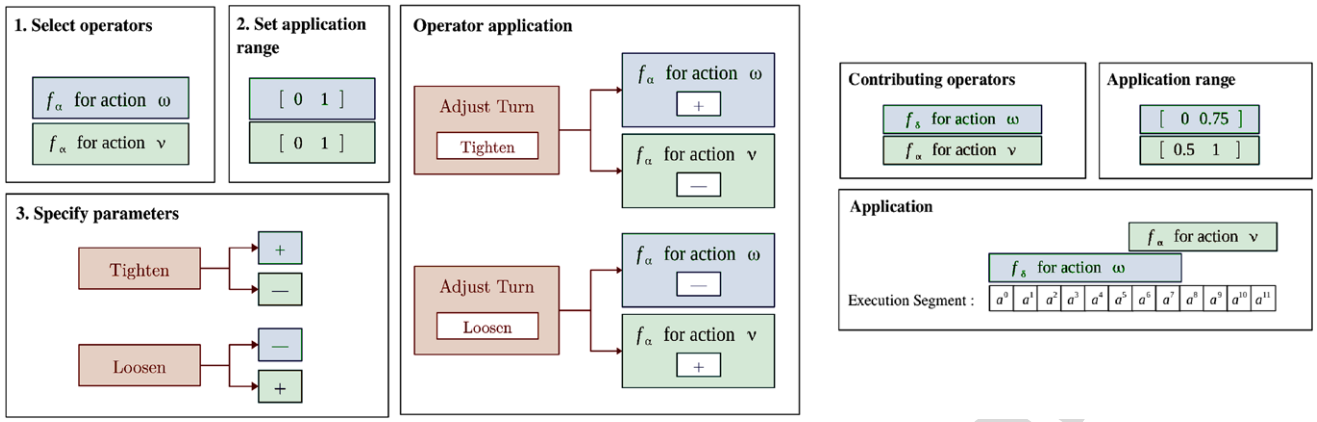

Fig. 5 *Left*: Illustration of the advice-operator building interface. *Right*: Example application range, over 12 datapoints

tion predictions, the baseline operators are defined as first- and second-order modifications to each individual action dimension. Concretely, given a segment $\hat{\xi}_d$ of a recorded observation-action execution trace ξ_d, such that subset $\hat{\xi}_d \subseteq \xi_d$ contains N_ξ observation-action pairs, $\hat{\xi}_d = \{(\mathbf{z}^i, \mathbf{a}^i)\}_{i=1}^{N_\xi}$, and each action is a vector in $\mathbb{R}^m$ with component values $a^i_j \in \mathbf{a}^i, a^i_j \in \mathbb{R}, i = 1..N_\xi, j = 1..m$, then the baseline operators are:

1. *Static* f_δ: A static modification amount δ_j, s.t. $f_\delta : a^i_j \rightarrow a^i_j + \delta_j$.
2. *Fractional* f_α: A fractional modification amount α, s.t. $f_\alpha : a^i_j \rightarrow a^i_j + \alpha \cdot a^i_j$.
3. *Incremental Fractional* f_β: Linearly increasing fractional modification amounts $\frac{i \cdot \beta}{N_\xi}$, s.t. $f_\beta : a^i_j \rightarrow a^i_j + \frac{i \cdot \beta}{N_\xi} \cdot a^i_j$.

Constant parameters $\alpha > 0, \beta > 0$ are set empirically by hand ($\alpha = \frac{1}{3}$ and $\beta = \frac{1}{2}$ in Sect. 5), and how to set $\delta_j > 0$ is described in Sect. 6.3. Note that there exists a unique parameter δ_j for each action dimension $j = 1..m$, while the parameters α, β are constant across all action dimensions. Each operator furthermore takes a *binary parameter*, indicating whether the modification amount should be positive or negative.

The first operator f_δ allows for the static augmentation of an action value, and is particularly useful if the value was previously zero or of opposite sign to the desired value. The second operator f_α enables augmentations that increase or decrease in proportion to the size of the executed action value, which is useful for producing smooth changes as well as very large or very small modification amounts. The reasoning behind the final operator f_β is to allow for incremental easing into a modification; useful, for example, when ramping speeds up/down when exiting/entering a turn.

6.2 Scaffolding Advice-Operators

Advice-operator development continues with the definition of complex operators. In particular, these operators are built, or scaffolded, from existing operators, beginning with the baseline set. Our empirical work provides an interface through which advice-operators are composed and sequenced into more complex operators. In brief, advice-operators are built as a hierarchy, or tree, of existing operators. Selection of an operator triggers an *ordered*[12] sequence of calls to underlying operators. The leaf nodes of this tree are the baseline operators, whose functioning is specified by the hand-written mathematical functions described in the previous section. The mathematical functioning of all non-baseline operators thus results from the sequencing and composition of functions built from the baseline mathematics.

The advice-operator building interface functions as follows (e.g. Fig. 5, left). In the first step, the ordered set of *child* operators that will contribute to the new *parent* operator is defined. In the second step, the range over which each child operator will be applied is indicated, as a fraction of the full segment (e.g. Fig. 5, right). This allows for flexibility in the duration of the contributing operators, where a given child operator can be applied over only a portion of the execution segment, if desired. In the third step, the input parameters for the new operator are specified. Recall that the baseline operators each take a binary parameter as input, to indicate whether the modification amount should be positive or negative. To now define a parameter for the parent operator, one parameter each is selected from the parameter lists of the contributing child operators, and composed into a single vector that constitutes a new parameter for the parent operator. Multiple parameter combinations may be specified for a given parent operator.[13]

[12] Note that operator composition is not transitive.

[13] The limit being the number of unique combinations of the parameters of the child operators.

Journal ID: 12369, Article ID: 156, Date: 2012-07-02, Proof No: 3, UNCORRECTED PROOF

6.3 Constraining and Conditionalizing Corrections

Data modification through advice-operators amounts to data *synthesis* from learner executions and teacher feedback. This synthesis is subject to multiple constraints intended to produce data that is firmly grounded on both the underlying learner execution, and the physical capabilities of the robot. Furthermore, steps are taken to ensure that data which corrects an iterative, developmental, policy does not negatively impact the final policy behavior.

The modification amounts produced by advice-operators are explicitly linked to the physical constraints of the robot. In particular, given a constant value $\gamma_j \in \mathbb{R}$ that describes the rate of change for dimension j of the action space (e.g. linear and rotational accelerations within our empirical validations), the static modification amount $\delta_j \in \mathbb{R}$ of operator f_δ is defined as $\delta_j \equiv \gamma_j \cdot dt$, where dt is the execution framerate.[14] The final advice-modified action is additionally constrained to lie within an amount $\eta_j \in \mathbb{R}$ of the executed action value. This constrains the synthesized points to lie near points that were actually executed, and thus near to the capabilities of the existing policy (which the robot is capable of executing). In particular, assuming a *maximum* rate change value of $\gamma_{j,max}$, the constraining amount η_j is defined as $\eta_j \equiv \gamma_{j,max} \cdot dt$.[15] This theoretically guarantees that the advice-modified datapoint is reachable from the executed point within one timestep.

One consequence of constraining datapoint modifications in this manner is that the mapping represented by the synthesized data might not be consistent with the behavior desired of the *final* policy. This is because the synthesized data is constrained to lie near the learner execution which, given that the execution was corrected, presumably was *not* consistent with the final desired behavior. Though the corrected, synthesized mapping does represent an *iterative* improvement on the mapping exhibited by the learner execution, this corrected mapping may still conflict with the desired behavior of the final policy. As an illustration, consider a learner executed translational speed of 1.2 $\frac{\text{m}}{\text{s}}$, a target behavior speed of 2.5 $\frac{\text{m}}{\text{s}}$, and a constrained modification amount of $\eta_j = 0.3$ $\frac{\text{m}}{\text{s}}$. A step in the direction of the target speed produces an advice-modified action with value 1.5 $\frac{\text{m}}{\text{s}}$; to add this data to the demonstration set equates to providing an example at speed 1.5 $\frac{\text{m}}{\text{s}}$, which is an improvement on the learner executed speed but is suboptimal with respect to the target behavior speed. The addition of this datapoint to the set thus amounts to providing the final policy with a suboptimal behavior example which, like any suboptimal behavior example, will degrade policy performance.

To circumvent this complication, our approach augments the state observation formulation with internal observations of the current action values. This anchors the action *predictions* of the observation-action mapping to the *current* action values (at the time of the state observation, prior to execution of the predicted action). Mappings now represent good behavior *given* the current action values, in addition to the current state observation values, and thus will not conflict with the final policy even if they are not good examples of the target final behavior. Returning to our illustration, under this formulation the speed 1.5 $\frac{\text{m}}{\text{s}}$ will be considered an appropriate prediction only when the current speed is around 1.2 $\frac{\text{m}}{\text{s}}$. Should the learner later revisit that area of the world with a speed of 2.0 $\frac{\text{m}}{\text{s}}$, for example, the policy will not attempt to slow the learner down to 1.5 $\frac{\text{m}}{\text{s}}$. The benefit of this observation formulation is more robust and flexible feedback-giving, since corrections that improve the behavior of an iterative policy, but are suboptimal for the final policy, are no longer a hazard. A drawback is that this observation formulation increases the dimensionality of the observation-space, which typically correlates to slower learning times and the need for more training data.

7 Future Directions

This section identifies promising research directions for the application and development of advice-operators and the F3MRP framework.

Our work has applied advice-operators to the wheel speeds of a differential drive robot, but low-dimensional action-spaces do not define the limit of advice-operators in general. The application of advice-operators to higher-dimensional action-spaces is not assumed to be straightforward, and may require additional translational techniques. For example, an advice-operator that corrects a high degree-of-freedom manipulator to be positioned "more to the right" might not operate directly in the action space of joint angles, but rather first in the 3-D spatial position of the end-effector, followed by a pass through an inverse kinematic controller to complete the translation to the action space. We identify the development of advice-operators for more complex spaces as a promising area, that furthermore is necessary to confirm the feasibility, or infeasibility, of advice-operators as policy correction tools in high-dimensional spaces.

The majority of our developed advice-operators to date have been *action*-modifying. Equally valid, however, are *observation*-modifying advice-operators. One such observation-modifying example is presented in the experiments of [2]: the observation features encoded the goal (target

[14] If a constant value for the rate of change in action dimension j is not defined for the robot system, reasonable options for this value include, for example, average rate of change seen during the demonstrations.

[15] The value $\gamma_{j,max}$ is defined either by the physical constraints of the robot, or artificially by the control system.

Journal ID: 12369, Article ID: 156, Date: 2012-07-02, Proof No: 3, UNCORRECTED PROOF

« SORO 12369 layout: Large v.1.3.2 file: soro156.tex (Andrius) class: spr-twocol-v1.4 v.2012/05/30 Prn:2012/07/02; 9:40 p. 12/13»
« doctopic: OriginalPaper numbering style: ContentOnly reference style: basic»

pose) of the task, and one of the operators functioned by recomputing these features with the executed value (actual final pose) as the target. The execution thus became a good behavior example, albeit for a different goal. This general idea—to encode some performance or goal metric, and then set it to match the executed value—is one option for the development of observation operators.

While the F3MRP framework was designed specifically for mobile robot applications, the techniques of this framework could apply to non-mobile robots through the development of an alternative mechanism for the selection of execution points by the teacher. For example, to visually display the 3-D Cartesian-space path taken by the end effector could serve as a suitable alternative for a robotic arm.

The policy update approach of this work simply adds new examples of good behavior, however an alternative could correct the actual points already in the dataset. The advantage would be to increase the influence of a correction and thus also the rate of policy improvement, since otherwise the undesirable data does remain in the set, degrading policy performance. Taking care to correct only *nearby* datapoints would be key to the soundness of this approach, so that points were not incorrectly adjusted. The F3MRP framework already computes a distance-based measure of dataset support; this measure also could be used to determine whether datapoints contributing to the prediction should be corrected or not.

8 Conclusions

The area of continuing to learn from feedback following demonstration has been the central topic of this article. Key considerations for the design of feedback types and interfaces were identified. The focus of this work has been corrective feedback from a human teacher, for use within mobile robot motion control domains. Our presented approach contributed most notably the corrective feedback form of advice-operators and the F3MRP interface for providing feedback, both of which have been designed to address challenges particular to low-level motion control with mobile robots. Our approach has been empirically validated in the form of multiple algorithmic variants, and corrective feedback has been shown to be an efficient and effective complement to demonstration. Open areas such as the application to more complex domains and non-mobile robots suggest rich directions for future research.

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. Abbeel P, Coates A, Quigley M, Ng AY (2007) An application of reinforcement learning to aerobatic helicopter flight. In: Proceedings of advances in neural information processing
2. Argall B, Browning B, Veloso M (2008) Learning robot motion control with demonstration and advice-operators. In: Proceedings of the IEEE/RSJ international conference on intelligent robots and systems
3. Argall B, Browning B, Veloso M (2009) Automatic weight learning for multiple data sources when learning from demonstration. In: Proceedings of the IEEE international conference on robotics and automation
4. Argall B, Browning B, Veloso M (2011) Teacher feedback to scaffold and refine demonstrated motion primitives on a mobile robot. Robot Auton Syst 59(3–4):243–255
5. Argall B, Chernova S, Veloso M, Browning B (2009) A survey of robot learning from demonstration. Robot Auton Syst 57(5):469–483
6. Atkeson CG, Moore AW, Schaal S (1997) Locally weighted learning. Artif Intell Rev 11:11–73
7. Atkeson CG, Schaal S (1997) Robot learning from demonstration. In: Proceedings of the fourteenth international conference on machine learning (ICML'97)
8. Bagnell JA, Schneider JG (2001) Autonomous helicopter control using reinforcement learning policy search methods. In: Proceedings of the IEEE international conference on robotics and automation
9. Bentivegna DC (2004) Learning from observation using primitives. Ph.D. thesis, College of Computing, Georgia Institute of Technology, Atlanta, GA
10. Billard A, Callinon S, Dillmann R, Schaal S (2008) Robot programming by demonstration. In: Siciliano B, Khatib O (eds) Handbook of robotics. Springer, New York, Chap. 59
11. Breazeal C, Scassellati B (2002) Robots that imitate humans. Trends Cogn Sci 6(11):481–487
12. Calinon S, Billard A (2007) Incremental learning of gestures by imitation in a humanoid robot. In: Proceedings of the 2nd ACM/IEEE international conference on human-robot interactions
13. Chernova S, Veloso M (2008) Learning equivalent action choices from demonstration. In: Proceedings of the IEEE/RSJ international conference on intelligent robots and systems
14. Eaton JW (2002) GNU Octave Manual. Network Theory Limited
15. Grollman DH, Jenkins OC (2007) Dogged learning for robots. In: Proceedings of the IEEE international conference on robotics and automation
16. Ijspeert AJ, Nakanishi J, Schaal S (2002) Learning rhythmic movements by demonstration using nonlinear oscillators. In: Proceedings of the IEEE/RSJ international conference on intelligent robots and systems
17. Kober J, Peters J (2009) Learning motor primitives for robotics. In: Proceedings of the IEEE international conference on robotics and automation
18. Kolter JZ, Abbeel P, Ng AY (2008) Hierarchical apprenticeship learning with application to quadruped locomotion. In: Proceedings of advances in neural information processing
19. Matarić MJ (2002) Sensory-motor primitives as a basis for learning by imitation: Linking perception to action and biology to robotics. In: Dautenhahn K, Nehaniv CL (eds) Imitation in animals and artifacts. MIT Press, Cambridge, Chap. 15

Journal ID: 12369, Article ID: 156, Date: 2012-07-02, Proof No: 3, UNCORRECTED PROOF

20. Nehaniv CL, Dautenhahn K (2002) The correspondence problem. In: Dautenhahn K, Nehaniv CL (eds) Imitation in animals and artifacts. MIT Press, Cambridge, Chap. 2
21. Nicolescu M, Mataric M (2003) Methods for robot task learning: Demonstrations, generalization and practice. In: Proceedings of the second international joint conference on autonomous agents and multi-agent systems
22. Pastor P, Kalakrishnan M, Chitta S, Theodorou E, Schaal S (2011) Skill learning and task outcome prediction for manipulation. In: Proceedings of IEEE international conference on robotics and automation
23. Peters J, Schaal S (2008) Natural actor-critic. Neurocomputing 71(7–9):1180–1190
24. Ratliff N, Bradley D, Bagnell JA, Chestnutt J (2007) Boosting structured prediction for imitation learning. In: Proceedings of advances in neural information processing systems
25. Smart WD (2002) Making reinforcement learning work on real robots. Ph.D. thesis, Department of Computer Science, Brown University, Providence, RI

Brenna D. Argall is the June and Donald Brewer Junior Professor of Electrical Engineering and Computer Science at Northwestern University. She holds a Faculty Research Scientist position at the Rehabilitation Institute of Chicago, where she is founder and director of a laboratory for rehabilitation robotics research. Prior to joining NU and RIC, she was a postdoctoral fellow (2009–2011) at the Swiss Federal Institute of Technology at Lausanne (EPFL). She received in 2009 her Ph.D. from the Robotics Institute at Carnegie Mellon University, where she also completed in 2006 a M.S. in Robotics and in 2002 a B.S. in Mathematics. Her research interests lie at the intersection of robotics, machine learning and rehabilitation, with a particular focus on learning from demonstration and human feedback.

Brett Browning is a Senior Systems Scientist at the Robotics Institute of Carnegie Mellon University. He is a member of the National Robotics Engineering Center, and also runs a research lab at Carnegie Mellon's Qatar campus. His research interests focus on robot perception, learning, and autonomy, with a focus on field robotics systems for industrial environments including Oil and Gas and mining operations. Browning received his Ph.D. in Computer Science and Electrical Engineering from the University of Queensland in 2000. He earlier received his Bachelor of Science and Bachelor of Electrical Engineering degrees in 1996 also from the University of Queensland.

Manuela M. Veloso is Herbert A. Simon Professor of Computer Science at Carnegie Mellon University. Veloso researches in artificial intelligence and robotics and founded and directs the CORAL research laboratory, for the study of intelligent agents that Collaborate, Observe, Reason, Act, and Learn, www.cs.cmu.edu/~coral. With her students, Veloso has developed teams of autonomous soccer robots, and service mobile robots in a symbiotic autonomy with humans and the web. Veloso is EEEE Fellow, AAAS Fellow, and AAAI Fellow. She is the President-Elect of AAAI and, for the last three years, she was the President of the RoboCup Federation, of which she continues to be a member of the Board of Trustees. Veloso was recently recognized by the Chinese Academy of Sciences as Einstein Chair Professor. She also received the 2009 ACM/SIGART Autonomous Agents Research Award for her contributions to agents in uncertain and dynamic environments, including distributed robot localization and world modeling, strategy selection in multiagent systems in the presence of adversaries, and robot learning from demonstration. Veloso is the author of one book on "Planning by Analogical Reasoning" and author in over 250 journal articles and conference papers.