Policy Adaptation through Tactile Correction

Brenna D. Argall and **Eric L. Sauser** and **Aude G. Billard** [1]

Abstract. Demonstration learning is a powerful and practical technique to develop robot behaviors. Even so, development remains a challenge and possible demonstration limitations, for example correspondence issues between the robot and demonstrator, can degrade policy performance. This work presents an approach for policy improvement through a *tactile* interface located on the body of the robot. We introduce the *Tactile Policy Correction (TPC)* algorithm, that employs tactile feedback for the *refinement* of a demonstrated policy, as well as its *reuse* for the development of other policies.

1 Introduction

Motion control is fundamental to many robotics applications, yet the development of motion behaviors remains a challenge. The development of control *policies*, that map world state to robot actions, typically requires a significant measure of effort and expertise. Moreover, many of the challenges associated with policy development only grow with domain and robot complexity, for example high degree of freedom humanoids. To refine a policy based on execution experience however can reduce the requirements placed on a developer, by increasing policy robustness. Some development effort also might be mitigated if a new policy is built by bootstrapping from an existing policy. The approach to policy development taken in this work is founded on both ideas, of policy refinement and reuse.

Under our approach a policy initially develops within a *Learning from Demonstration (LfD)* paradigm. Under LfD, teacher executions of a desired behavior are recorded and a policy is derived from the resultant dataset. LfD has seen success on a variety of robotics applications, and has the attractive characteristic of being an intuitive means for human teacher to robot learner knowledge transfer, as well as being an accessible policy development technique for those who are not robotics-experts. For a full review of LfD design decisions, we refer the reader to [2] and [5].

Even with the advantages secured through demonstration, policy development typically is still a non-trivial task. Our algorithm incorporates human feedback in order to further reduce the strain placed on the policy developer. Feedback is used for policy updating in two capacities (Sec. 1.1), and is provided through a tactile interface n the robot body (Sec. 1.2).

1.1 Policy Refinement and Reuse

To have a robot learn from its execution performance, or *experience*, is a valuable policy improvement tool for any development technique. Within the context of LfD specifically, execution experience can be used to overcome limitations in the demonstration dataset. One typical limitation is dataset sparsity, since demonstration from every world state is infeasible in all but the simplest domains. Other limitations include the issue of correspondence between the teacher and learner, and deficiencies in the performance of the teacher, who may in fact provide suboptimal or ambiguous demonstrations. Overcoming potential dataset limitations is key to good policy performance, since a LfD policy depends heavily on the quality of the demonstration data from which it is derived.

A variety of routes may be taken to incorporate information gathered from experience into a LfD algorithm, and thus *refine* the resultant policy. For example, execution experience is used to update state transition models [1] and state values [12]. Other approaches provide more demonstration data, driven by learner requests for more data [7, 8] or more teacher-initiated demonstrations [6].

Given the challenge of developing robust control policies, the ability to *reuse* existing policies, designed to address related tasks, is a practical feature for any policy learning system. Policy reuse under LfD occurs most frequently in the form of behavior primitives, or simpler policies that contribute to the execution of a more complex policy. Examples include hand-coded primitives used within [11] or automatically extracted from [4] demonstrated tasks, and primitives learned from demonstration [3]. In this work we consider the approach of policy *bootstrapping*; that is, of building a new policy from an existing policy able to accomplish a *different* task. Moreover, similar characteristics between the tasks are automatically extracted for reuse by our approach.

1.2 Tactile Corrections

The algorithm presented in this paper uses corrective feedback from a human teacher to refine and reuse policies. A fundamental consideration for such an approach is the mechanism used to provide feedback to the learner; for example, feedback might be provided verbally to indicate successful task completion [9]. The feedback form we explore is this work is that of *tactile corrections*.

We posit that tactile feedback furthers one of the strengths of demonstration-based learning. Namely, humans use touch to instruct other humans in certain contexts, for example when demonstrating a pose or motion that requires a particular position or trajectory in 3-D space, like a ballet posture or tennis swing. Moreover, one of the attractive features of demonstration-based policy development is accessibility to those who are not robotics experts. Policy development by non-experts strongly suggests robot operation around humans, in which case t The detection of tactile interactions can be crucial for safe robot operation. These tactile detections may be exploited to further knowledge transfer from human to robot.

A handful of works within the field of *Human-Robot Interactions (HRI)* utilize human touch for the development of robot behaviors. Tactile feedback is detected in order to minimize the support forces provided by a teacher during humanoid behavior learning [10], and

[1] École Polytechnique Fédérale de Lausanne (EPFL), Lausanne, Switzerland, email: [brennadee.argall, eric.sauser, aude.billard]@epfl.ch

behavior selection is adapted by using tactile reward signals [13]. Under our framework, tactile feedback provides a *correction* on the policy execution. We consider correcting poor policy predictions to be a particularly direct approach to addressing potential LfD limitations, since it provides not only a performance evaluation, but also an indication of a more suitable alternate prediction.

Policy correction has seen limited attention within LfD. Most approaches have a human teacher indicate the correct prediction from a discrete set of actions with significant time duration [7, 11], though some work provides corrections within continuous state-action domains sampled at a rapid rate [3]. Our work targets similarly operates within rapidly sampled continuous state-action spaces, targeting low-level motion control domains.

2 The Tactile Policy Correction Algorithm

We introduce *Tactile Policy Correction (TPC)* as an algorithm for the refinement and reuse of motion policies, accomplished via tactile feedback from a human teacher.

Under TPC, a policy is initially derived from demonstrations of a task by a teacher. We formally define the world to consist of actions $A \in \mathbb{R}^{\ell}$ and observations $Z \in \mathbb{R}^{(m+n)}$ of world state. An observation $\mathbf{z} \in Z$ consists of two components, $\mathbf{z} = (\mathbf{z}_{\varphi}, \mathbf{z}_{\neg\varphi})$, where $\mathbf{z}_{\varphi} \in \mathbb{R}^m$ describes the robot pose, and $\mathbf{z}_{\neg\varphi} \in \mathbb{R}^n$ describes any other observables that are of interest to the policy.[2] We define a *demonstration* to consist of a sequence of observations $\{\mathbf{z}^j\}_{j=1}^{T}$, recorded during teacher execution of the task. The collected set $D = \{\mathbf{z}^j\}_{j=1}^{N}$ of demonstrations, totaling N recorded observations, is then provided to the robot learner. From this set a policy $\pi : Z \rightarrow A$ is derived, that enables the selection of an appropriate action given the observed state.

Following demonstration and policy derivation, the robot executes with its policy and receives tactile corrections from the human teacher. Tactile corrections are used within two capacities, either to refine the existing policy or to build a new policy bootstrapped on the demonstrated policy. Pseudo-code for this approach is provided in Algorithm 1.

Algorithm 1 *Tactile Policy Correction*

```
1: Given D
2: initialize w ← 0, D_c ← {}
3: derive π ← policyDerivation(D, D_c, w)
4: while correcting do
5:    initialize φ̃^0 ← 0, z^0 ← (z^0_φ = φ^0, z^0_¬φ)
6:    for t = 1 to T do
7:       Policy π execution:
8:          predict  φ̂^t ← regression(z^{t-1})
9:          adjust   φ̂^t ← φ̂^t + φ̃
10:         execute  φ^t ← controller(φ̂^t)
11:      if detect touch ϑ^t then
12:            map      δ^t ← M(ϑ^t)
13:            correct  φ^t ← controller(φ^t + δ^t)
14:            record   φ̃^t ← φ̃^{t-1} + δ^t
15:      end if
16:      update  z^t ← (z^t_φ = φ^t, z^t_¬φ)
17:      record  D_c ← D_c ∪ z^t
18:   end for
19:   update  w
20:   rederive π ← policyDerivation(D, D_c, w)
21: end while
22: return π
```

[2] Pose information is necessary for the TPC algorithm, and so $\mathbf{z}_{\varphi} \neq \emptyset$. The presence of additional observation information however is application-dependent, and possibly absent such that $\mathbf{z}_{\neg\varphi} = \emptyset$.

2.1 Algorithm Execution

The first phase of the TPC algorithm consists of task demonstration by the teacher, producing dataset D from which the learner derives an initial policy π. The second phase of the algorithm involves learner execution with the policy π, and corrective tactile feedback. If during this phase the policy is used to accomplish the demonstrated task, then *refinement* of the demonstration policy is the result. If instead the policy is used to accomplish a different, *undemonstrated*, task, then a new policy is the result and policy *reuse* has been accomplished. In either case, this second phase of the algorithm consists of learner execution with the policy π, followed by teacher correction and a policy update to incorporate the correction. This *execution-correction-update* cycle continues to the satisfaction of the teacher.

A single execution-correction-update cycle is presented in lines 5-20 of Algorithm 1. Policy execution (lines 7-10) at timestep t consists of two phases: prediction of a target pose $\hat{\boldsymbol{\varphi}}^t$, and the selection of an action $\mathbf{a}^t \in A$ to accomplish that pose. Pose prediction is accomplished via regression techniques, based on state observation $\mathbf{z}^{t-1}$ (lines 8). Action selection is accomplished via a robot-specific controller, and its execution results in a new robot pose $\boldsymbol{\varphi}^t$ (line 10).

The human teacher may choose to offer a tactile correction at any timestep of an execution. If detected, the robot learner translates the tactile feedback $\boldsymbol{\vartheta}^t$ into an incremental shift $\boldsymbol{\delta}^t \in \mathbb{R}^m$ in the robot pose, according to the defined mapping M (line 12). The form taken by the feedback $\boldsymbol{\vartheta}^t$ is platform-specific, depending both on the tactile sensors employed to detect human touch and how the sensor feedback is processed.

The robot controller is then passed the new *adjusted* pose, for which the incremental shift $\boldsymbol{\delta}^t$ is added to the current robot pose $\boldsymbol{\varphi}^t$ (line 13). The influence of this incremental shift is maintained over multiple timesteps, through an offset parameter $\tilde{\boldsymbol{\varphi}}^t \in \mathbb{R}^m$ that maintains a sum of all adjustments seen during the execution (line 14) and is added to the pose prediction at each execution timestep (line 9).

The timestep concludes with the recording of observation $\mathbf{z}^t$, into the set of corrected execution points D_c (line 17). The tactile correction thus also is recorded, since the current pose $\boldsymbol{\varphi}^t$ has been corrected by tactile feedback and this corrected pose is recorded into observation $\mathbf{z}^t$ through component $\mathbf{z}^t_{\varphi}$.

Upon completion of the entire execution, policy π is rederived from demonstration set D *and* the set of corrected executions D_c (lines 20); the corrected execution thus is treated as a new demonstration for the policy. Policy derivation furthermore gives greater importance to the corrected data, through weight w (Sec. 2.2.2).

Important to note is that the TPC algorithm is agnostic to the techniques used for pose prediction (`regression`) and action selection (`controller`) during policy execution, as well as to the technique that translates tactile feedback into a pose adjustment (mapping M). The following section describes the formulation for the mapping M in our initial implementation of the TPC algorithm.

2.2 Tactile Corrections

The tactile interface of our empirical implementation consists of *Ergonomic Touchpads*[3] located on the manipulator arm. The pads detect contact presence and relative motion, which we map to a change in end-effector position and orientation (Sec. 2.2.1). The movement that results then is recorded into the demonstration dataset and incorporated into a policy update (Sec. 2.2.2).

[3] `http://www.ergonomictouchpad.com/`

2.2.1 *Online Modification of Policy Execution*

Four touchpads, $T_0 \cdots T_3$, encircle the lower forearm of the robot arm (near the wrist), and one, T_4, is located on the back of the robot hand (Fig. 1a). In practice, we decompose the mapping $M : \boldsymbol{\vartheta} \rightarrow \boldsymbol{\delta}$ into two distinct parts which operate separately on the wrist (end-effector) and hand of the robot arm (Fig. 1b), which proved an intuitive mapping in practice for the human user providing corrections.

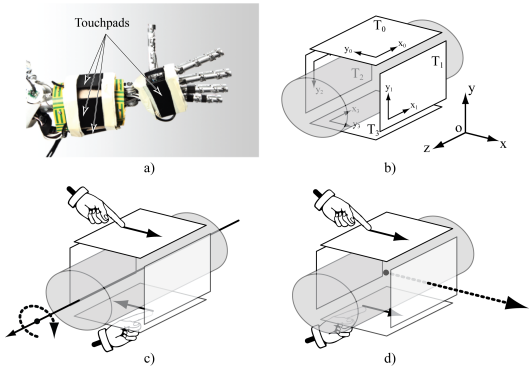

Figure 1. a) Setup of the touchpads on the wrist and hand of the robot. b) Schematic of the pads controlling the wrist. c,d) A sliding movement on opposite pads results in either rotational or translational arm motion.

The first part operates on the first 5 DoF leading to the wrist of our 7-DoF manipulator. The pads $T_{\{0..3\}}$ can be seen as an interface for controlling the manipulator within the 6-DoF Cartesian space (position and orientation). As shown in Figure 1 (c,d), sliding the fingers along two opposite touchpads can lead to either a translational or rotational motion command, depending on the directional congruency between the motion of the fingers. When motion is detected on the touchpads, the commands are resolved by an inverse kinematic controller which moves the joints of the robot accordingly.

The second part of the mapping relates to the final 2 DoF controlling the robot hand. Since the last pad has 2 DoF as well, the mapping of motion commands is one to one. Finally, the target pose adjustement $\boldsymbol{\delta}^t$ is computed by taking the pose $\boldsymbol{\varphi}^t$ of the robot end-effector after correction and by substracting the pose $\hat{\boldsymbol{\varphi}}^t$ predicted by the model to it.

Tactile feedback from the touchpads is somewhat limited in comparison to more sophisticated tactile sensors, for example that provide force information or a finer spatial resolution. As a result, in practice corrective repositioning is not always as responsive as the teacher requires. Therefore, our implementation pauses policy execution while a target adjustment is being offered to the learner, such that lines 12-13 of the algorithm psuedo-code loop until corrective positioning is complete. Note however that this limitation results from a deficiency in our *hardware*, rather than in the algorithm. We plan to validate TCP on a more sophisticated tactile sensor in future work.

2.2.2 *Incorporation into a Policy Update*

Upon completion of an execution corrected with tactile feedback, new data is incorporated into the policy. Policy execution in the TCP algorithm consists of a pose prediction via regression techniques, followed by action selection by a controller. Under our implementation the controller is statically defined, and so policy derivation (Alg. 1, `policyDerivation`) consists of regression parameter estimation only. Policy derivation gives greater importance to the corrected data, through a weight w^j

$$w^j = \begin{cases} \left(1 - \frac{N}{N+N_c}\right)(1 - \bar{w}(\tau) & \mathbf{z}^j \in D \\ \left(1 - \frac{N_c}{N+N_c}\right)\bar{w}(\tau) & \mathbf{z}^j \in D_c \end{cases}$$

for point $\mathbf{z}^j \in D \cup D_c$, where N is the number of datapoints in D, and N_c the number in D_c, and $\bar{w}(\tau)$ is a global weight function, that in our implementation depends on execution time τ which is encoded in $\mathbf{z}^j_{\neg\varphi}$. Note that to preferentially weight the new data effectively amounts to *forgetting* the behavior of the original policy. It therefore is not expected that the behavior of a policy reused within the TCP algorithm will continue to be expressed after tactile guidance to produce an alternate behavior.

3 Discussion and Conclusion

We have introduced *Tactile Policy Correction (TCP)* as an algorithm for the refinement and reuse of policies through tactile feedback from a human teacher. With tactile corrections, we aim to mitigate some potential limitations in demonstration-based learning. A first validation of TCP, on a small 57-DoF humanoid learning to position the end-effector of its 7-DoF arm for the grasping of different objects at various locations, has shown promising results that will be published in later work.

ACKNOWLEDGEMENTS

This research was funded by EU Projects IST-2004-004370 (RobotCub), IST-04169 (feelix-growing) and FET-7 grant number 231500 (RobotSkin).

REFERENCES

[1] Pieter Abbeel and Andrew Y. Ng, 'Exploration and apprenticeship learning in reinforcement learning', in *Proceedings of ICML*, (2005).
[2] Brenna Argall, Sonia Chernova, Brett Browning, and Manuela Veloso, 'A survey of robot learning from demonstration', *Robotics and Autonomous Systems*, **57**(5), 469–483, (2009).
[3] Brenna D. Argall, *Learning Mobile Robot Motion Control from Demonstration and Corrective Feedback*, Ph.D. dissertation, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, March 2009.
[4] Darrin C. Bentivegna, *Learning from Observation Using Primitives*, Ph.D. dissertation, College of Computing, Georgia Institute of Technology, Atlanta, GA, July 2004.
[5] Aude Billard, Sylvain Callinon, Rudiger Dillmann, and Stefan Schaal, 'Robot programming by demonstration', in *Handbook of Robotics*, eds., B. Siciliano and O. Khatib, Springer, New York, NY, USA, (2008).
[6] Sylvain Calinon and Aude Billard, 'Incremental learning of gestures by imitation in a humanoid robot', in *Proceedings of HRI*, (2007).
[7] Sonia Chernova and Manuela Veloso, 'Learning equivalent action choices from demonstration', in *Proceedings of IROS*, (2008).
[8] Daniel H. Grollman and Odest Chadwicke Jenkins, 'Dogged learning for robots', in *Proceedings of ICRA*, (2007).
[9] Andrea Lockerd and Cynthia Breazeal, 'Tutelage and sociallyl guided robot learning', in *Proceedings of IROS*, (2004).
[10] Takashi Minato, Yuichiro Yoshikawa, Tomoyuki Noda, Shuhei Ikemoto, Hiroshi Ishiguro, and Minoru Asada, 'CB2: A child robot with biomimetic body for cognitive developmental robotics', in *Proceedings of IROS*, (2007).
[11] Monica N. Nicolescu and Maja J. Mataric, 'Methods for robot task learning: Demonstrations, generalization and practice', in *Proceedings of AAMAS*, (2003).
[12] Martin Stolle and Christopher G. Atkeson, 'Knowledge transfer using local features', in *Proceedings of ADPRL*, (2007).
[13] Kazuyoshi Wada and Takanori Shibata, 'Social effects of robot therapy in a care house - change of social network of the residents for two months -', in *Proceedings of ICRA*, (2007).