Article

The International Journal of Robotics Research
2019, Vol. 38(14) 1560–1580

DOI: 10.1177/0278364919871998
journals.sagepub.com/home/ijr
SAGE

Reinforcement learning of motor skills using Policy Search and human corrective advice

Carlos Celemin[1,2], Guilherme Maeda[3,4], Javier Ruiz-del-Solar[1], Jan Peters[5] and Jens Kober[2]

Abstract

Robot learning problems are limited by physical constraints, which make learning successful policies for complex motor skills on real systems unfeasible. Some reinforcement learning methods, like Policy Search, offer stable convergence toward locally optimal solutions, whereas interactive machine learning or learning-from-demonstration methods allow fast transfer of human knowledge to the agents. However, most methods require expert demonstrations. In this work, we propose the use of human corrective advice in the actions domain for learning motor trajectories. Additionally, we combine this human feedback with reward functions in a Policy Search learning scheme. The use of both sources of information speeds up the learning process, since the intuitive knowledge of the human teacher can be easily transferred to the agent, while the Policy Search method with the cost/reward function take over for supervising the process and reducing the influence of occasional wrong human corrections. This interactive approach has been validated for learning movement primitives with simulated arms with several degrees of freedom in reaching via-point movements, and also using real robots in such tasks as "writing characters" and the ball-in-a-cup game. Compared with standard reinforcement learning without human advice, the results show that the proposed method not only converges to higher rewards when learning movement primitives, but also that the learning is sped up by a factor of 4–40 times, depending on the task.

Keywords

Reinforcement learning, policy search, learning from demonstrations, interactive machine learning, movement primitives, motor skills

1. Introduction

In Learning from Demonstrations (LfD) (Argall et al., 2009; Chernova and Thomaz, 2014), a human teacher provides examples of the task execution to an agent or robot, using either teleoperation, kinesthetic teaching, or by providing demonstrations with his or her own body. Those demonstrations are recorded and used to compute a model of the primitive intended for executing the task. With LfD, the robot can quickly learn a policy that satisfies simple tasks, or in complex cases, a policy that is close to its satisfaction. However, LfD has a number of limitations: (1) it requires expert demonstrations; (2) the learned skill depends on the quality of the demonstrations provided by the users; (3) some of the demonstrations might be suboptimal or ambiguous. This constrains the performance of the learned policy.

Robot motor skill learning has been the subject of research for several years. Machine learning has been used for obtaining trajectory representations, since several applications face the necessity of encoding sequences of points into a policy model that can be used to execute a motor skill. Motor primitives models have been used to represent the movements required for many tasks, including

[1]Department of Electrical Engineering & Advanced Mining Technology Center, University of Chile, Chile
[2]Cognitive Robotics Department, Delft University of Technology, Netherlands
[3]Preferred Networks, Inc., Japan
[4]Department of Brain Robot Interface, ATR Computational Neuroscience Lab, Japan
[5]Intelligent Autonomous Systems lab, Technische Universität Darmstadt, Germany

Corresponding author:
Carlos Celemin, Cognitive Robotics Department, Delft University of Technology, Building 34, Mekelweg 2, 2628 CD Delft, Netherlands.
Email: c.e.celeminpaez@tudelft.nl

drumming (Pongas et al., 2005), T-ball batting (Peters and Schaal, 2008), the ball-in-a-cup game (Kober and Peters, 2009), pancake flipping (Kormushev et al., 2010), ball throwing, dart throwing, robot table tennis (Kober et al., 2012), and executing a golf swing (Maeda et al., 2016).

The mentioned tasks have been solved using policies based on models that present some generalization capabilities, such as the Dynamic Movement Primitives (DMPs) model (Ijspeert et al., 2002), primitives based on Gaussian mixture models (Khansari-Zadeh and Billard, 2011), or probabilistic movement primitives (ProMPs) (Paraschos et al., 2013). These models have interesting properties that can be convenient in many applications. Movement primitives can be learned through demonstrations or by self-improvement using reinforcement learning.

In robotics, reinforcement learning (Kober et al., 2013) has been used to learn and improve movement primitives, often initially acquired from demonstrations. Policy Search methods have shown to be particularly suitable for learning with real robots (Deisenroth et al., 2013), attaining higher rewards with respect to an initial (demonstrated) policy. However, the optimization process requires a large number of trials, which is usually impractical or expensive when using real systems. Also, the Policy Search method strongly relies on good demonstrations, since it is a local search method (Deisenroth et al., 2013), but for certain tasks the human teacher might not be able to provide useful demonstrations, particularly when the dynamics of the task or the limitations of the robot are unknown. For example, how many swings are required to achieve a ball-in-a-cup task, given a heavy ball and a robot with limited accelerations that cannot toss the ball high enough in one shot? This lack of intuition suggests some form of interactive learning process where human knowledge is added or transferred as the robot optimizes the policy.

In this work, we propose a method for learning motor skills with real robots, which renders convergence feasible in very few episodes. It combines Policy Search algorithms with human corrections in order to leverage the exploration provided by the Policy Search method with the knowledge of a human teacher. Our proposed method guides the exploration of a Policy Search algorithm with the human teacher's current knowledge of the task. We assume this process to be dynamic in the sense that the teacher knowledge also improves on observing the effects of the corrections through the interaction of the robot with the environment and its respective performance.

The corrections advised by the teachers are in the actions domain, since it fits better to Policy Search methods (actor-only); in previous works it has been shown that the corrective advice in this domain fits better with continuous action problems, and that non-expert users can lead the learning agents to high-performance policies, even though they cannot teleoperate the agent properly to fulfill the task (Celemin and Ruiz-del Solar, 2018). Moreover, human teachers like to provide feedback in this domain more than in the evaluative domain. It has been shown that users prefer to give information about "how to execute the task" or "how to improve the task" than "how good is an action or a policy" (Amershi et al., 2014; Suay et al., 2012; Thomaz et al., 2006). Therefore, the framework COACH (Corrective Advice Communicated by Humans) (Celemin and Ruiz-del Solar, 2018) is used to incorporate the teacher corrections in the Policy Search loop, in which the human teacher occasionally advises corrections during execution time. The advice is a relative change of the magnitude of the action executed at time step t. The advice is given to the agent during execution but in the time step $t+1$, which is immediately after the execution of the action to be corrected.

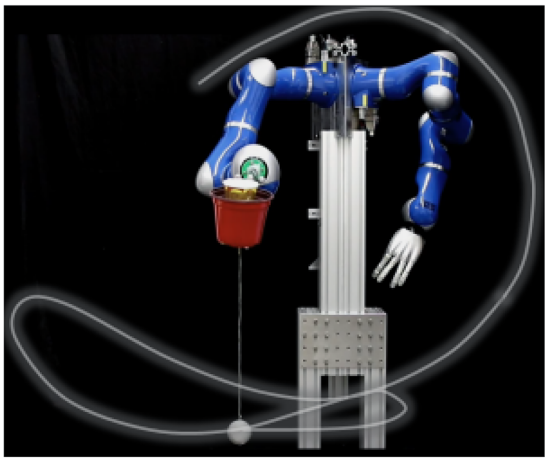


Fig. 1. Ball-in-a-cup task execution of a policy learned using the proposed interactive Policy Search method.

Additionally, this paper introduces a method to adapt COACH for training policies in problems where the teacher's corrections are in the task domain, but the policy computes actions in the joint space. This mismatch between the reference frames of actions and human corrections is known as the "correspondence problem" (Argall et al., 2009; Breazeal and Scassellati, 2002; Schaal et al., 2003). The mapping of the human corrective feedback in the task domain onto the the joint space allows one to effectively train policies for robot arms of several degrees of freedom (DoF).

The experiments presented in this paper show that the proposed method can be utilized to shape detailed trajectories with vague action corrections. The human corrective feedback for shaping trajectories is tested in a problem of learning to write characters with simulated and real robot arms. Significant reductions in the number of trials required to learn reaching movements with simulated arms are presented; actually the introduced method is tens of times faster than the conventional Policy Search method when the arm has 50 DoF. The method is also tested in the real-world problem of the ball-in-a-cup game (Figure 1). In this case, successful policies can be obtained four times faster than with the traditional Policy Search approach.

The remainder of this paper is organized as follows. Section 2 presents related work. Section 3 provides the

background of the methods proposed in this work. Section 4 introduces the use of corrective feedback to shape movement primitives. The Policy Search method with human corrective advice, which is the main contribution of this work, is presented in Section 5. In Section 6, the validation experiments and the results are presented. Finally, the conclusions are discussed in Section 7.

2. Related work

This section presents a brief introduction to interactive machine learning methods, along with their application for training movement primitives. The closest approaches to the method proposed in this work are described and their limitations are listed in order to motivate the presented contribution.

2.1. Interactive machine learning

Considering the drawbacks already mentioned about LfD, learning approaches that address the participation of human teachers throughout the entire robot learning process—as opposed to only the initial demonstration—-have been proposed. When observing a current suboptimal policy execution, the teacher obtains some insights about policy enhancement, and participates in the learning loop by interactively providing feedback during policy learning, in order to perform corrections.

The teacher's feedback can be evaluative. In these approaches, the teacher evaluates how desirable an executed action is, through signals of reward or punishment. Interactive reinforcement learning (Thomaz and Breazeal, 2006), TAMER (Knox and Stone, 2009), and the work of Thomaz and Breazeal (2007) have been used to explore the considerations of this kind of human feedback in contrast to the reward function of an autonomous reinforcement learning scheme. Moreover, human teachers can assess policies in *learning from human preferences* approaches, in which the teacher iteratively observes the execution of two different policies and chooses the one that is considered best (Akrour et al., 2011, 2014), with good results in simple tasks; this approach has also been applied to the learning of complex simulated tasks with deep neural networks (Christiano et al., 2017), and applied to manipulation tasks with real robots (Jain et al., 2013).

The feedback given by a human teacher can be corrective in the actions domain. The agent can be interrupted by the user while executing a policy, in order to make improvements (Meriçli et al., 2011). The user provides demonstrations for the current state, and the new data are attached to the policy, in order to be executed in similar states. For tasks of continuous actions, the Advice-Operator Policy Improvement (A-OPI) approach (Argall et al., 2008) is a framework that allows human teachers to provide corrections on relative changes to executed actions (e.g., relative change of action magnitude). COACH (Celemin and Ruiz-del Solar, 2018) is a framework based on the same kind of feedback used by A-OPI but, additionally, it is intended to be used at execution time, since it has a module that models the human teacher's intentions, to adapt the size of the policy correction, and a module that handles the delay of the human response.

2.2. Interactive corrections with movement primitives

Interactive corrections can also be applied in the context of movement primitives. In this case, corrections are used to update the parameters that shape the characteristics of the robot movement. The feeding assistance robot developed by Canal et al. (2016) is preprogrammed with a ProMP (Paraschos et al., 2013) for feeding disabled people. Then a framework is proposed to allow caregivers to personalize the original trajectory to the preferences of the disabled person. In this framework, the caregiver physically adapts the ProMP execution through kinesthetic feedback. The new executed path is recorded to create a new ProMP. Argall et al. (2011) proposed a system that allows the modification of a primitive during execution with tactile feedback. If the user provides a correction with an effector displacement with respect to the original trajectory, the displacement is applied from there on to the rest of the path, then all the data points are recorded to rederive the policy after the execution.

Kinesthetic teaching is used for incremental refinement of trajectories of context-dependent policies (Ewerton et al., 2016), represented with ProMPs, wherein the user may modify the trajectory execution to perform a correction. Then the data points of the recorded trajectory are applied to update the probability distribution of the ProMP. That method was tested in reaching tasks with a robot arm.

2.3. Interactive corrections via coarse feedback

There are a number of cases where none of the previous approaches can actually be applied. Here we describe a few scenarios:

1. There is no expert available to provide high-quality demonstrations that lead to a policy with acceptable performance (e.g., a person with a disability without a caregiver, who needs to fix a policy for a new task or environment).
2. The final user does not have access to an interface to provide new demonstrations, such as intuitive teleoperation interfaces, wearable sensors, or motion capture systems.
3. The robot is not back-driveable or its dimensions are not suited for direct human physical interaction required by kinesthetic teaching.
4. The task involves fast robot movements, making kinesthetic corrections impractical or unsafe for the teacher.

In these cases, wherein detailed feedback cannot be provided to the learner, approaches like A-OPI or COACH that are based on simpler signals of correction are better suited. Since there is no possibility of detailed corrections, these algorithms are suitable because in this case the human-agent interface does not need tactile or force sensors, and can be limited to simple, sparse, occasional, and vague commands given via a keyboard, voice commands, or gestures.

The approach by Schroecker et al. (2016) is closest to our work; in that approach, the problem caused by the absence of expert demonstrations was partially overcome by combining a Policy Search algorithm with interactive definitions of via-points for adapting DMPs. In that work, the teacher could stop the trajectory execution and move a robot (physically or remotely) to a desired position at that specific time step. This via-point correction was then used to update the distribution used for the Policy Search exploration. The method was validated in simulations of writing letters and object insertion with a robot arm. The approach by Schroecker et al. (2016), however, only addressed the kinematics of the tasks, focusing on the shape of the trajectories. This interactive Policy Search strategy is not suitable for tasks where the dynamics are dominant (e.g., throwing and catching an object) since stopping the task to adapt a specific via-point is physically impractical and the instant of correction is not evident. Also, modifying a via-point to satisfy certain kinematic configurations invariably affects the acceleration and thus the outcomes of a dynamic task.

In this paper, we use COACH (Celemin and Ruiz-del Solar, 2018) as the mechanism for interactive corrections as it allows human feedback to be introduced *during* robot execution. However, COACH was originally designed for interactive optimization of policies in a Markov decision process setting. In this paper, we provide a new formulation of COACH for time-dependent parametrized policies that makes it compatible with the improvement of movement primitives.

3. Background

The learning method proposed in this paper is a hybrid algorithm that employs the interaction of a human teacher for guiding the convergence of a Policy Search algorithm through the COACH framework. This section briefly introduces the basics of the Policy Search method, followed by the COACH framework.

3.1. Policy Search

Known as an actor-only method, the Policy Search method is a class of reinforcement learning algorithms that learn parametrized policies directly in the parameter space using a cost or reward function. Unlike critic methods, these algorithms do not compute a value function. Computing the value function requires observing transitions in the complete state-action space, leading to a heavy demand of data and time that is not feasible with most real physical systems. Therefore, in robotic applications, the Policy Search method is often more suitable than value-function-based reinforcement learning. Additionally, the use of parametrized policies reduces the search space, which is important where there are time and energy limitations, as is usually the case of learning with robots (Deisenroth et al., 2013).

Algorithm 1. Model free Policy Search.

1: **repeat**
2: **Explore**: Execute M roll-outs using π_k
3: **Evaluate**: Obtain outcomes of trajectories or actions
4: **Update**: Compute π_{k+1} given the roll-outs and evaluations
5: **until** Policy converges $\pi_{k+1} \approx \pi_k$

Moreover, in robotic applications, the Policy Search method is a better choice than value-based methods, owing to the properties of scalability and stability of the convergence (Kober et al., 2013), because a small change of the policy may lead to large changes of the value function, which in turn can produce large changes of the policy. Given unlimited training on simulated environments, this high sensitivity to parameter changes may be convenient for finding the globally optimal solution. Real robots, however, demand stable, smooth, and fast convergence.

Typically, a Policy Search method works with three steps, as shown in Algorithm 1: exploration, evaluation, and updating. The exploration step creates samples of the current policy for executing each roll-out or episode. In the evaluation step, the quality of the executed roll-outs is assessed, i.e., the exploration done during the roll-out execution is evaluated with the cost or reward function. The update step uses the evaluation of the roll-outs to compute the new parameters of the policy. This update can be based on policy gradients, expectation-maximization, information theoretic, or stochastic optimization approaches.

During the last years, several Policy Search algorithms have been proposed and evaluated using different strategies in each of the three steps. For the case of policies represented as movement primitives, well-known algorithms exist, such as Policy Learning by Weighting Exploration with the Returns (PoWER) (Kober and Peters, 2009), Relative Entropy Policy Search (Peters et al., 2010), Policy Improvement with Path Integrals (PI^2) (Theodorou et al., 2010), and black-box optimization methods, such as Covariance Matrix Adaptation Evolutionary Strategy (Hansen and Ostermeier, 2001; Metzen et al., 2014; Stulp and Sigaud, 2013). Our proposed method can work with any Policy Search variant by accessing the exploration noise and adapting it with human feedback.

3.2. COACH: Corrective Advice Communicated by Humans

With COACH (Celemin and Ruiz-del Solar, 2018), a human teacher provides occasional binary feedback as a

correction in the action domain, in order to update the current policy for the state wherein the advised action is executed. The trainer has to advise the correction immediately after the execution of the action to be modified. The binary signals increase or decrease the magnitude of the executed action, and this signal can be given independently for every degree of freedom that composes the action vector. The interactive learning is based on the four modules reviewed next.

3.2.1. Human feedback modeling. $H(\boldsymbol{s})$ is a module learned during the training process that tries to predict the human advice regarding the current state $\boldsymbol{s}$. The model is applied to compute the adaptive learning rate used for updating the parameters of the policy model $P(\boldsymbol{s})$. That prediction defines how much the policy will be modified when the teacher advises a correction in the visited state $\boldsymbol{s}$.

3.2.2. Human feedback supervised learner. This module updates the parameters $\boldsymbol{v}$ of the human feedback module $H(\boldsymbol{s}) = \boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{v}$, using a stochastic gradient descent based strategy. These weights are updated proportionally to the value of the prediction error, i.e., the difference between the human advice h and the prediction given by $H(\boldsymbol{s})$. The state space is mapped to the features space onto the features vector $\boldsymbol{\Phi}$, which is the same used in the human feedback module $H(\boldsymbol{s})$ and the policy module $P(\boldsymbol{s})$.

3.2.3. Policy supervised learner. The policy $P(\boldsymbol{s})$ is the module that maps the observed states onto actions $(P(\boldsymbol{s}) = \boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{w})$. In advising a correction, the human teacher is trying to modify the executed $P(\boldsymbol{s})$; therefore, this module updates the weighting vector of the policy model $(\boldsymbol{w})$, also using a stochastic gradient descent based strategy. But in this case, the error prediction is unknown, since the teacher provides the correction trend, not the exact value of the action to be executed. Because of this error assumption, COACH sets the prediction error as

$$\text{error} = h \cdot e \tag{1}$$

where h and e are the sign and magnitude of the error respectively, h is the teacher feedback (-1 or $+1$; that is, decrease or increase), and e is a constant value defined at the beginning of the learning process.

When a large change must be applied to the magnitude of the action at a state $\boldsymbol{s}$, the teacher would advise a sequence of corrections of constant sign (either only $+1$ or only -1); these corrections in the *human feedback module* make the absolute value of the prediction $H(\boldsymbol{s})$ tend asymptotically to 1, which is the largest possible magnitude. Conversely, when the user is fine-tuning the magnitude of the action with small changes, the sequence of advised corrections would alternate between -1 and $+1$, which makes the absolute value of $H(\boldsymbol{s})$ tend to zero. This correlation of $|H(\boldsymbol{s})|$ and the size of the action modification

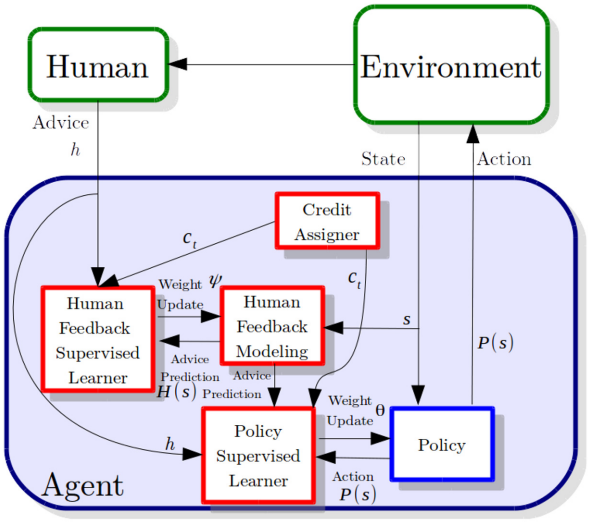


Fig. 2. General scheme of COACH, with its internal module.

intended by the teacher can be used to detect when to apply small or large updates to the policy by means of an adaptive learning rate based on this prediction. The adaptive learning rate or size step of the policy is calculated from $|H(\boldsymbol{s})|$, the absolute value of H in each time step.

3.2.4. Credit assigner. This module is necessary for problems of high frequency, in which human teachers are not able to advise each executed continuous action at every independent time step. The credit assigner module tackles this problem by associating the feedback not only to the last state-action pair, but to several past state-action pairs. Each past state-action pair is weighted with the corresponding probability that characterizes the human response delay. In this process, a new feature vector is computed by the credit assigner $(\boldsymbol{\Phi}^{\text{cred}})$; this is the actual vector used by the human feedback and the policy supervised learner modules. This module is borrowed from TAMER, which introduces a gamma distribution based on psychological studies of human response to events of different complexities. That distribution is often approximated by a uniform distribution in its support region, so it prunes time steps with low probability; more details can be found in the paper by Knox and Stone (2009).

The general scheme of COACH is presented in Figure 2, wherein the connections between the aforementioned modules are depicted.

4. Corrective advice for shaping movement primitives

This section proposes modifications of the COACH algorithm for training parameterized movement primitives. The original COACH algorithm was proposed for a Markov decision process setting (Celemin and Ruiz-del Solar, 2018), where human feedback is used to improve a policy

that evolves under the Markov assumption in a state space $\boldsymbol{s}$. In this article, the goal is to adapt COACH for training movement primitives, so that human feedback can be used to correct a time-dependent policy. This work is focused on movement primitives, such as DMPs or ProMPs, whose evolutions are defined by a phase variable z_t, but other representations are also possible (Khansari-Zadeh and Billard, 2011). Essentially, the representation changes from the state variable $\boldsymbol{s}$ to the phase variable z_t.

In DMPs, the policy is represented by a dynamic system comprising a linear spring damper (equation (2)), determined by the constants α_f and β_f, and attached to a goal attractor x_{goal}

$$f_t = \alpha_f(\beta_f(x_{\text{goal}} - x_t) - \dot{x}_t) \tag{2}$$

This system is modified by an arbitrary nonlinear term $\mathbf{g}(z_t)^\top \boldsymbol{w}$ for shaping complex trajectories such that its acceleration is

$$\frac{1}{\tau}\ddot{x}_t = f_t + \mathbf{g}(z_t)^\top \boldsymbol{w} \tag{3}$$

where $\boldsymbol{w}$ is the parameters vector (or the weighting vector), $\mathbf{g}(z_t)$ is the basis function vector, and τ is the sampling time of the system.

In the case of ProMPs, the policy model is a probability distribution in the parameter space, represented with a mean and a covariance matrix obtained from a set of demonstrations. During execution, this probability distribution is conditioned for every specific context, so that the model applies the Bayesian rule of conditioning to compute the most probable trajectory, given an observation. With the Bayesian inference, a set of parameters $\boldsymbol{w}$ of a linear model of basis functions is obtained and the trajectory is computed using

$$x_t = \mathbf{g}(z_t)^\top \boldsymbol{w} \tag{4}$$

COACH can be used to train the movement primitive by updating the parameter vector $\boldsymbol{w}$ via human advice. The update uses stochastic gradient descent with the error assumption in equation ((1)), and the derivative

$$\frac{\partial x_t}{\partial \boldsymbol{w}_l} = [\mathbf{g}_t]_l \tag{5}$$

where the right-hand side of the equation represents the lth basis function at time step t. Since this work is focused on correcting single trajectories, in the cases of ProMPs, the corrected vector $\boldsymbol{w}$ can be used to update the global mean and covariance matrices, as in Ewerton et al. (2016).

The user advises local corrections during the motor skill execution. This advice propagates over the next time steps as changes in the weightings are smoothed ahead by the shape of the basis functions $\mathbf{g}_t$ (usually radial basis functions), so that the effect of the correction can be appreciated immediately.

While COACH can be used to advise corrections in either the task or joint space of the robot, advising corrections at the joint level is not intuitive for human demonstrators who teach robots with many DoF. Thus, hereinafter, this work addresses the case where corrections are made in the task space of the robot. In other words, we assume that the human demonstrator is not going to advise corrections to the robot's joint movements. Under this assumption, the algorithm still needs to be considered for policies for computing actions in either the task or joint domains. In the second case especially, the algorithm has to deal with the correspondence problem between corrections in the end-effector space mapped to the joint space. If the policy is represented at the task level in Cartesian space, the COACH adaptation is straightforward and an inverse kinematics function must be used as a last step to map the learned policy to the joint space of the robot, as is typically done for this kind of policy. Conversely, if the policy is represented in the joint space, the inverse kinematics function must be used on the human feedback to translate the human correction into the corresponding space.

For policies that represent the end-effector trajectory, the learning scheme is simpler because, according to our assumption, human corrections are in the same domain, so they can be used to modify the policy directly. Policies in the Cartesian domain can be used only with robots that have an operation mode that uses an inverse model, so the learning scheme requests commands in the end-effector space and maps them to actions in joint space, e.g., an inverse kinematics that translates position requests to the respective vectors of joint angles.

Figure 3 depicts the scheme for this kind of policy. The left-hand side shows the stage in which the demonstrations are gathered and the initial policy is obtained; the right-hand side shows the stage of learning with COACH wherein the human teacher advises the executed action. The "Update" block computes the COACH modules and modifies the parameters vector of the policy; the blocks surrounded by red dashed lines work during and after the learning process for executing the current policy. Between the policy computation and the action execution, there is a block for mapping the action computed by the policy onto the actuator space.

Algorithm 2 presents the general COACH framework for training movement primitives represented with a linear combination of basis functions. Functions like *computePolicy*() or *updatePolicy*() perform different computations, depending on whether the policy represents actions in task or joints space.

We first explain the algorithm considering the details for training policies in the Cartesian space; thereafter, we introduce the differences for the case of policies in the joint space, wherein the "correspondence problem" needs to be solved. The algorithm begins by stating some variables and defining the magnitude for the error assumption e in equation (1) and the learning rate β (lines 1 and 2). The n weightings c_t that represent the probability function of the

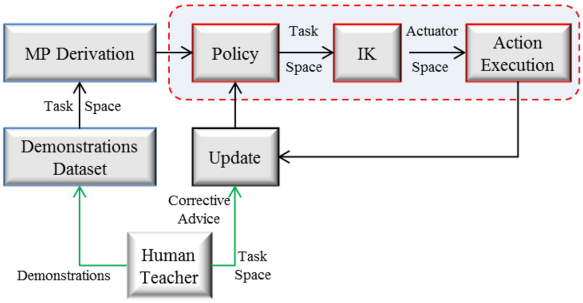


Fig. 3. Learning scheme for movement primitives in the Cartesian space.
IK: inverse kinematics; MP: movement primitive.

credit assigner are computed (lines 3 and 4). The loop between lines 5 and 19 is executed once per time step. The function *getBasisFunctions*(z_t) maps the phase variable to the features vector $\mathbf{g}_t$ (line 6); in line 7, the function *computePolicy*(w, g_t) computes the action based on the linear model of the policy (equation (4)), which is used internally by the inverse model to compute the action q_t in joint space; then the action is executed by the robot (line 8). If human advice h is received before the next time step (line 10), the steps in lines 12 to 19 are executed. When there is a human correction, the features vector that includes the weighted sum of the past features vectors $\mathbf{\Phi}^{\text{cred}}$ is computed (lines 13 and 14); afterwards, the expected human feedback prediction $H(z_t)$ is computed (line 15) and its parameters $\boldsymbol{v}$ updated with the stochastic gradient descent (SGD) rule (line 16); the adaptive learning rate $\alpha(z_t)$ is obtained with an additional bias that avoids situations where learning rates are almost zero, which would preclude changes in the policy, after the corresponding human correction (in practice, we set bias = 0.05) (line 17). The error assumption from equation (1) is computed (line 18), and named $\boldsymbol{error}_X$, where the subscript X means that this error is defined in Cartesian space. Then the policy module is updated in a similar way as the human feedback modules, except using the $\boldsymbol{error}_X$ assumption, so that the function *updatePolicy*($error_X, \Phi^{\text{cred}}, w$) computes Δw with

$$\Delta w \leftarrow error_X \cdot \Phi^{\text{cred}} \tag{6}$$

and updates the weightings of the policy.

4.1. Solving the correspondence problem for policies defined in the joint space

Policies that map the actions in the joint space directly are simpler to execute as they are already given as actuator commands. However, during learning, more steps are required for solving the "correspondence problem" between the human advice and the actuator space. Figure 4 shows the scheme for learning this kind of policy, in which the user advises the correction in the task space, an inverse

Algorithm 2. COACH for training movement primitives.

1: $e \leftarrow$ constant
2: $\beta \leftarrow$ constant
3: **for** $0 \leqslant t < n$ **do**
4: $\quad c_t \leftarrow assignCredit(t)$
5: **for all** z_t **do**
6: $\quad g_t \leftarrow getBasisFunctions(t)$
7: $\quad q_t \leftarrow computePolicy(w, g_t)$
8: $\quad takeAction(q_t)$
9: $\quad$ wait for next time step
10: $\quad h \leftarrow getHumanCorrectiveAdvice()$
11: $\quad$ **if** $h \neq 0$ **then**
12: $\quad\quad \Phi^{\text{cred}} \leftarrow 0$
13: $\quad\quad$ **for** $0 \leqslant i < n$ **do**
14: $\quad\quad\quad \Phi^{\text{cred}} \leftarrow \Phi^{\text{cred}} + (c_i \cdot g_{t-i})$
15: $\quad\quad H(z_t) \leftarrow \Phi^{\text{cred}^{\text{T}}} \cdot v$
16: $\quad\quad \Delta v \leftarrow \beta \cdot (h - H(z_t)) \cdot \Phi^{\text{cred}}$
$\quad\quad v \leftarrow v + \Delta v$
17: $\quad\quad \alpha(z_t) \leftarrow |H(z_t)| + \text{bias} = |\Phi^{\text{cred}^{\text{T}}} \cdot v| + \text{bias}$
18: $\quad\quad error_X \leftarrow h \cdot e \cdot \alpha(z_t)$
19: $\quad\quad updatePolicy(error_X, \Phi^{\text{cred}}, w)$

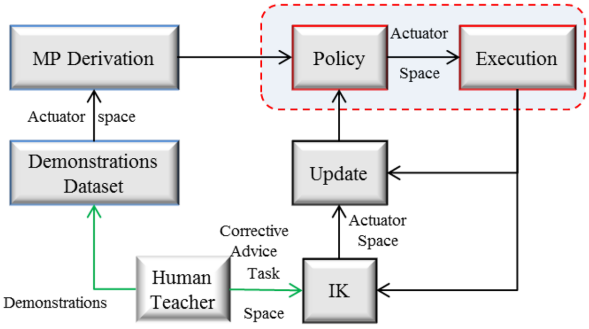


Fig. 4. Learning scheme for movement primitives in the joint space.
IK: inverse kinematics; MP: movement primitive.

kinematics block propagates this correction to the joints space, and the updating block modifies the weightings $\boldsymbol{w}$ based on the modules of COACH. During policy execution (the area surrounded by the red dashed line), the inverse model is not needed because it is used only during the time steps advised by the teacher in the learning process.

In contrast to the original COACH scheme, and the one used for policies in the Cartesian space, wherein the human model and the policy are in the same domain, here the human model H is in the task domain, whereas the policy model is in the joint space. Therefore, some additional steps are necessary to compute the correction of the policy.

In this case, in Algorithm 2 the function *computePolicy*(w, g_t) computes the action q_t to be executed directly, without the need of an inverse model, and until line 18 the implementation is the same. The function *updatePolicy*($error_X, \Phi^{\text{cred}}, w$) is the most important change, since it is responsible for mapping the correction to the joint space.

Algorithm 3. Function $updatePolicy(error_X, \Phi^{\text{cred}}, w)$ for policies in joint space.

1: **def** *update Policy*$(error_X, \Phi^{\text{cred}}, w)$
2: $q_t \leftarrow \Phi^{\text{cred}^{\text{T}}} \cdot w$
3: $\tilde{X}_t \leftarrow FK(q_t) + error_X$
4: $\tilde{q}_t \leftarrow IK(\tilde{X}_t, q_t)$
5: $error_q \leftarrow \tilde{q}_t - q_t$
6: $\Delta w \leftarrow error_q \cdot \Phi^{\text{cred}}$
7: $w \leftarrow w + \Delta w$

Algorithm 3 describes this function in which $\boldsymbol{error}_X$ is mapped to the policy domain using forward and inverse kinematics models. In line 2, the vector $\boldsymbol{q}_t$ is not computed with the current basis functions vector $\mathbf{g}_t$, but with the basis functions given by the credit assigner, i.e., the policy computes the expected action to be corrected according to the used human delay probability distribution. The current effector pose is computed with $FK(q_t)$ and added to $\boldsymbol{error}_X$ in order to obtain the "desired" effector position (line 3), which is used to obtain the "desired" joint vector based on the inverse kinematics model (line 4). In line 5, the difference between the "desired" joint vector and that computed by the current policy is considered the propagation of the error from the task space to the actuator space, called $\boldsymbol{error}_q$. Finally, the SGD is computed to update the weights $\boldsymbol{w}$ in lines 6 and 7.

5. Simultaneous corrective advice and Policy Search method for learning movement primitives

The learning methods presented in the previous section are fully based on human corrections. This can be useful in some simple problems, but in some others, the inherent drawbacks of interactive learning mentioned in Section 2.1 might influence the convergence more negatively. This section presents a core contribution of this work—a synergistic combination of the Policy Search method with human corrections—in the sense that the Policy Search method is used to reduce the impact of erroneous human feedback, while correct human feedback is used to speed up the learning process of a Policy Search algorithm.

The proposed method combines the evolution of Policy Search algorithms with the knowledge that a human teacher can share based on corrective advice. According to the cost function, the Policy Search algorithm decreases or "filters out" the influence of improper corrections given by the teacher. In essence, our proposed method enables human guidance based on COACH to influence and bias the exploration of a Policy Search algorithm in the form of exploration noise. Figure 5 depicts the proposed scheme. An initial parameter vector $\boldsymbol{w}^{\text{init}}$ is disturbed, either with the original exploration strategy defined by the Policy Search algorithm or with human guidance, through the roll-out execution. The parameter update is carried out according to the particular Policy Search implementation. These iterations continue until convergence, resulting in a final policy $\boldsymbol{w}^{\text{final}}$. As a result, this combination enables joint skill learning, where the robot can start with a blank policy, and human exploration is added whenever the human teacher judges that his or her knowledge on the task can benefit the robot learning.

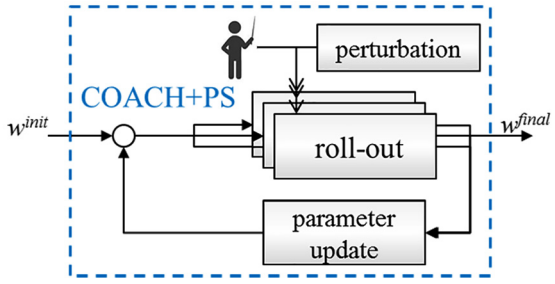


Fig. 5. Interactive Policy Search with COACH-based exploration.

Thus, during the roll-outs selected to be advised by the user, the COACH algorithm is run, but taking into account that the vector $\boldsymbol{w}$ is loaded into a $\boldsymbol{w}_t$ vector every time step in order to have all the changes in memory, since the evaluation stage (Algorithm 1, line 3) can be step-based, depending on the Policy Search algorithm used as the baseline (Deisenroth et al., 2013).

Algorithm 4 is a high-level description of the proposed strategy for complementing the Policy Search method with human advice during the exploration stage. Like Algorithm 1, the explore, evaluate, and update stages are run every kth iteration, but with an important difference in the exploration process. There are two exploration modes: the original exploration strategy of the Policy Search method, and the exploration based on COACH. The user chooses one of them through the flag *HumanGuidance* (line 4). If it is considered necessary and possible to advise the agent, the roll-out using the COACH-based exploration is run (line 5); otherwise, the teacher allows random exploration of the original Policy Search algorithm (lines 8 and 9). This condition is kept constant for the entire roll-out, so both sources of exploration are not combined. This allows the user to transparently observe the effect of the advice on the policy.

The flag *HumanGuidance* can be switched differently depending on the algorithm implementation. For instance, the human–machine interface can query it before every roll-out execution. In particular, for the implementations of this work, this flag is set *False* by default and switched on when the user advises a correction. If, during a roll-out, the user does not give corrective feedback, the flag is set *False* for the following roll-out.

During each iteration of the Policy Search algorithm, M roll-outs of T time steps are carried out; the vector $\boldsymbol{w}^{[1]}$ corresponds to $\boldsymbol{w}^{[\text{init}]}$ in Figure 5. Vector $[\boldsymbol{w}_m]_t$ contains the parameters $\boldsymbol{w}$ at time step t of the mth roll-out. For the roll-outs in which the user is giving corrective feedback

Algorithm 4. Policy Search with simultaneous human guidance.

1: **repeat**
2: **Explore**: first roll-out with the current policy π_k
$w_1 = w^{[k]}$
3: **for** $m = 1 \ldots M$ **do**
4: **if** HumanGuidance==True **do**
5: $[w_m]_t \leftarrow RunRollOut_{COACH}([w_m]_1)$
6: $[w_{m+1}]_1 \leftarrow [w_m]_T$
7: **else**
8: $[w_m]_t \leftarrow PS\_exploration(w^{[k]})$
9: $RunRollOut([w_m]_t)$
10: **Evaluate**: cost of each roll-out
$[R_m]_t = \phi_T + \sum_{t=0}^{T} r_t^{[m]}$
11: **Update**: Compute new policy parameters using
$w^{[k+1]} \leftarrow Update([w_m]_t, [R_m]_t)$
12: **until** Policy converges $\pi_{k+1} \approx \pi_k$

(*HumanGuidance*==True), the initial parameters vector used for the $(m+1)$th roll-out (line 6) is the last one resulting from the immediate previous roll-out, i.e., $[\boldsymbol{w}_m]_T$. This is to keep the same policy that is being incrementally advised by the teacher. The rest of the algorithm follows the regular Policy Search scheme.

6. Experiments and results

The use of human corrective feedback is first validated for shaping trajectories without the use of the Policy Search method. Thereafter, more complex experiments are presented to compare the proposed interactive Policy Search method with respect to the conventional Policy Search method.

6.1. Learning movement primitives with corrective advice

Experiments were carried out exclusively to evaluate the use of COACH for training movement primitives. As a proof of concept, we propose the problem of teaching a robot how to write letters, in simulation and using a real robot. The objective in this experiment is to evaluate improvements that can be made to the shape of a trajectory (encoded as a movement primitive) via corrective advice. This improvement is quantified against the original set of points that compose a letter, used as a ground truth. Two different approaches were evaluated: policy refinement, and policy reuse. In the first case, the goal is to improve the shape of a given letter. The latter case addresses an application of transfer learning, where the goal is to reuse one of the existing policies, and reshape it via corrective advice to fit a new desired symbol.

The procedure consisted of an initial stage, where a user interface was used to visually indicate a reference letter to be drawn on a screen. The user then provided a set of

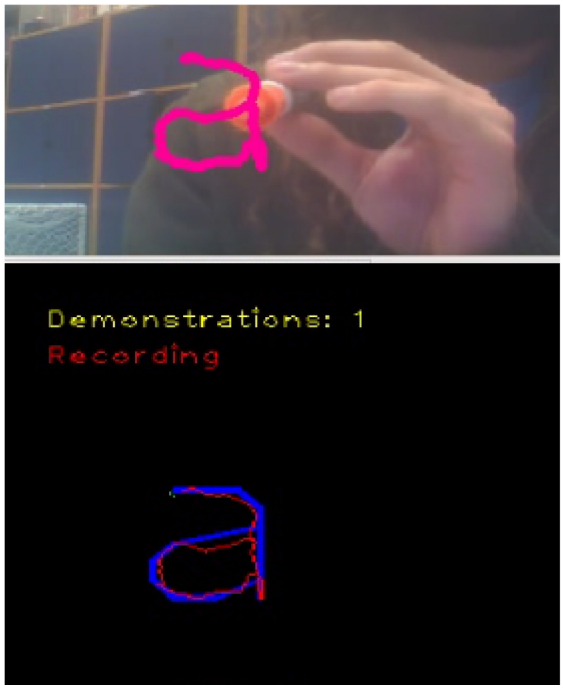


Fig. 6. Demonstration recording.

demonstrations of trajectories for each indicated letter. An RGB camera captured the user's pen movement and recorded the whole path into a dataset, which was used to train an initial Cartesian policy, parametrized as a ProMP (equation (4)) but without using the covariance. Figure 6 shows the screen of the interface for recording demonstrations. The top figure shows one instance of a human demonstration. The bottom figure shows the reference symbol (in blue) overlaid with one of the provided demonstrations (in red). Extension 1 shows the interface while demonstrations are recorded.[1]

In a second stage, the user attempted to refine the policy resulting from the first stage. Two different interactive approaches were used to correct the policy: (i) corrections with more demonstrations and (ii) corrections with COACH. To quantify the performance of both strategies, the learned symbols were compared with the ground truth symbols using the Euclidean distance after alignment with dynamic time warping.

The experiments were carried out both in simulation, with a 3-DoF robot arm, and with a real UR5 robot with 6-DoF. Figure 7 shows the simulated case, where the robotic arm draws the learned symbol (in red) while the teacher provides corrective feedback to correct the trajectory toward the ground truth reference (in blue). In each experiment, five participants demonstrated and corrected the robot primitive. The participants were between 26 and 37 years old; there were three engineers and two with a background in the social sciences. Each user took a first session of practice to become habituated to the interaction with the recording and corrective interfaces.

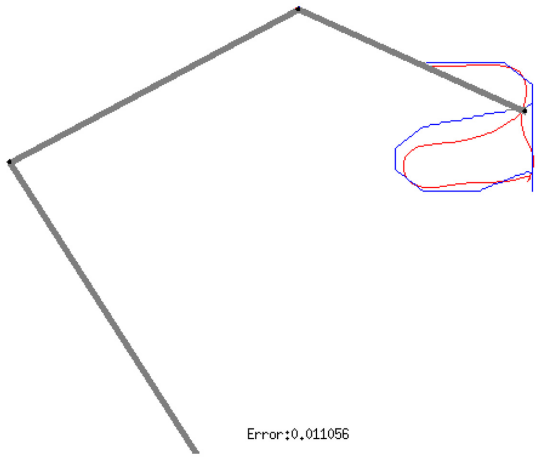


Fig. 7. Policy execution and correction with the simulated 3-DoF arm. Human feedback was used to make the robot draw as close as possible to the reference letter (in blue). The initial demonstration is shown in red.

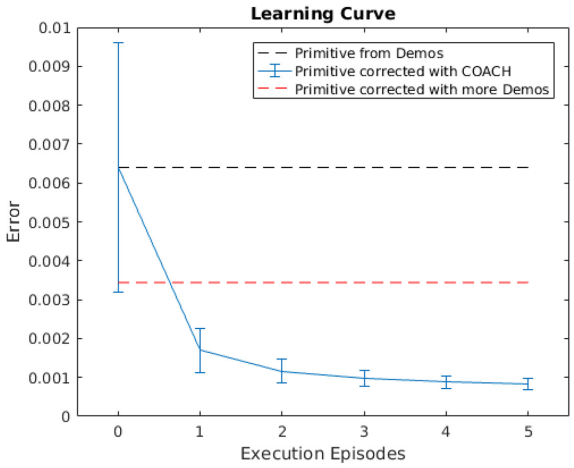


Fig. 8. Average error during learning to write with a simulated arm.

6.1.1. Experiments of policy refinement. In this experiment, a set of six symbols was learned (letters: a,c,I,m,p,s). The objective was to improve and refine the trajectories learned from demonstrations. For each symbol, three policies derived from the first set of demonstration are compared.

- *Primitives resulting from the original demonstrations.* For each letter, a policy was learned with a regression using a batch of five demonstrated trajectories.
- *Primitives resulting from corrections with COACH.* During five sequential executions, the users observed the policy execution resulting from the five demonstrations and simultaneously interacted to provide corrections using COACH. The corrections were relative to the original policy in the Cartesian space. The users used a keyboard with two keys for correcting along the x axis and two keys for correcting along the y axis. The users advised corrections to make the end effector pass closer to the reference symbol.
- *Primitives resulting from corrections with more demonstrations.* The users observed the initial policy from the five demonstrations, and provided five additional demonstrations for improvement.

Results of learning with a simulated arm. The average learning curves for all the symbols are plotted in Figure 8. In the cases of learning only with demonstrations, the figure plots a constant dashed line of the final error resulting after the regression with the datasets. Since the symbols used as reference are sets of points without physical dimensions, the demonstrations recorded in the pixel space are normalized, so the error measurements do not have units.

It is possible to see that when correcting the trajectory with COACH, providing corrective feedback during five episodes of the trajectory, the error is decreased by 79.83%. Conversely, the strategy of correcting with more demonstrations only obtained around 40% error reduction with respect to the primitives learned with the first demonstration dataset. The aforementioned reductions mean that with the same amount of trials (five new demonstrations vs. five episodes of correction with COACH), with corrective feedback, a human teacher can achieve almost twice the error reduction with respect to the strategy of recording more demonstrations.

Moreover, from the learning curves, it is possible to demonstrate that, with only one episode of corrective advice with COACH, the human teacher can attain better improvement than can be achieved with the new set of five demonstrations. After the first episode of advising corrections, the average error of the trajectories is decreased by 73%. Most of the improvement is achieved in the first episode. This observation is not only taken from the learning curves, but also from the appearance of the learned letters. For instance, Figure 9 shows the progress of improving a trajectory with corrective advice; most of the change obtained from the corrections happens in the first episode.

Table 1 shows the final errors obtained after correcting the policies with COACH and with more demonstrations for each of the six explored letters. Basically, the trends of the average results are kept; only one anomaly is highlighted. In the case of the letter "p", after the process of correction by recording more demonstrations, the error was increased by 2.41%. A correcting session with the simulated arm is shown in Extension 2.[2]

Results of learning with a real arm. The previous experiments were replicated using a real UR5 arm. The same symbols and comparisons were used in this case. The

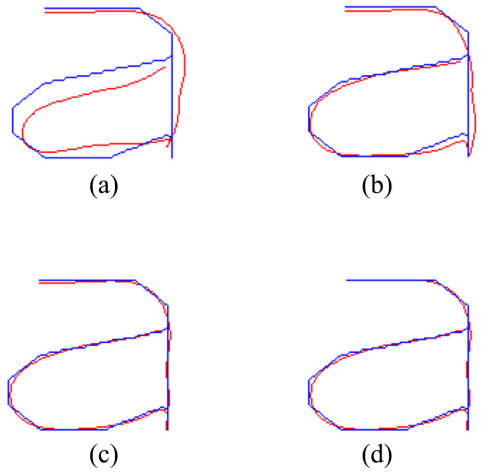


Fig. 9. Trajectory correction progress: learned primitive (red), symbol of reference (blue). (a) Policy without corrections, (b) after one correction episode, (c) after five episodes, (d) after eight episodes.

Table 1. Average error for each symbol trained, multiplied by 10^{-2}.

Symbol	Demos	Corrections (COACH)		Corrections (more demos)	
	Error	Error	Decreased error, %	Error	Decreased error, %
a	1.322	0.115	91.27	0.638	51.75
c	0.709	0.085	88.05	0.338	52.29
I	0.125	0.033	73.50	0.041	66.93
m	0.878	0.088	89.93	0.458	47.85
p	0.182	0.089	51.05	0.187	−2.41
s	0.432	0.064	85.17	0.300	30.48

participants were between 23 and 29 years old, only three with engineering skills. The points composing the drawn trajectories were obtained from the robot's odometry, and compared with the reference symbol for the error calculation.

Examples of corrected symbols are shown in Figure 10, where the initial policy obtained from demonstrations is drawn in white, while the final trajectory after five episodes of correction is shown in red. Figure 11 shows the average curves of learning and correcting with demonstrations in contrast with the learning curve of correcting with COACH. The error reduction obtained using more demonstrations is, on average, 30.7%, and around 84.4% when using COACH. In this case, the error reduction with the first episode of correction is 53.8%, which again is higher than the one achieved with five new demonstrations.

The final results for each symbol are listed in Table 2. These results have similar trends to the results with the

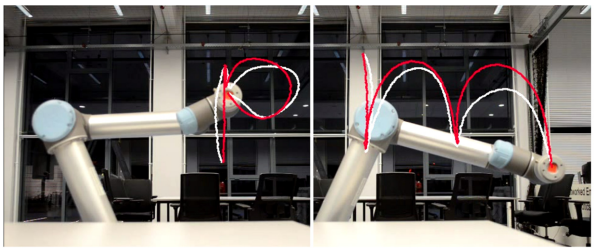


Fig. 10. Examples of learning the letters "p" and "m" with the real robot. Policy derived from demonstrations (white), and policy trained with corrective advice (red).

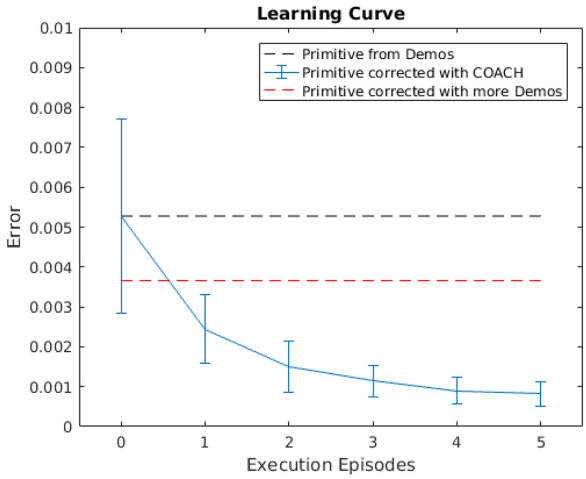


Fig. 11. Average error when learning to write with a real arm.

simulated arm. The lowest error reduction is with the symbol "p", but this is still higher than the average of the error reductions resulting from the strategy of correcting with more demonstrations. In general, these results are consistent with the previous ones. The use of corrective advice to shape trajectories is hence a good strategy for learning agents from human teachers, especially in situations wherein the combination of user expertise and the quality of the user interface does not obtain the best conditions for recording high-performance demonstrations. The results show that detailed trajectories can be shaped easily only using vague binary corrective feedback. A correcting session with the real UR5 arm is shown in the Extension 3.[3]

Figure 12 depicts the evolution of the amount of corrections advised by the teachers during the learning process. Every event of pressing a keyboard to suggest a change in the trajectory is counted as a correction. The results show that, in general, the evolution of the corrections is similar in both the simulated and the real arm experiments. Most (70%) of the corrections are given in the first and second episodes, wherein above 90% of the policy improvement is obtained.

Table 2. Average error for each symbol trained using the real robot. The error is multiplied by 10^{-2}.

Symbol	Demos	Corrections (COACH)		Corrections (more demos)	
	Error	Error	Decreased error, %	Error	Decreased error, %
a	1.483	0.158	89.35	1.151	22.45
c	0.842	0.115	86.34	0.401	52.38
I	0.284	0.096	66.20	0.182	36.62
m	1.287	0.127	90.13	0.967	24.86
p	0.174	0.109	37.36	0.159	8.62
s	1.206	0.216	82.09	0.796	34.00

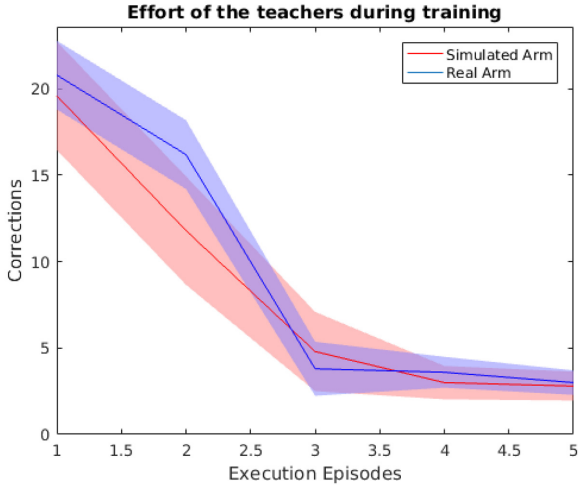


Fig. 12. Average of the corrections advised by the teachers for correcting the letters.

6.1.2 Experiments of transfer learning for policy reuse with a simulated arm. In policy reuse, the user can provide corrections to a primitive whenever a task is changed, or when the task has to be performed in a new environment. There can be cases in which recording demonstrations can be complicated, for different reasons, e.g., (1) the absence of an expert user in the task, who is able to provide high-quality demonstrations that lead to a policy with acceptable performance; (2) when the final user does not have access to an interface, such as wearable sensors or a complex vision system, to provide new demonstrations; (3) when kinesthetic teaching is not possible, since it is constrained to robots with physical dimensions that a human teacher can handle. For some of those cases, the knowledge already represented by the primitive can be reused; then the user only needs to execute local modifications for the points of the trajectory that need to be fixed for the new conditions. This previous discussion motivates the evaluation approach of policy reuse; this is presented alongside the results of policy refinement with a simulated robotic arm.

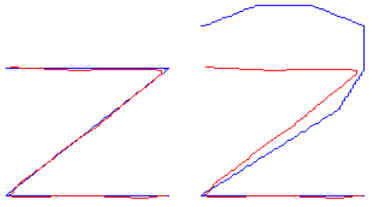


Fig. 13. Initial trajectory for the transfer learning process: from "z" to "2".

Here, users had to correct and improve a trajectory to reduce the error between the path printed by the simulated arm and the reference symbol taken as ground truth. In contrast to the previous experiments of policy refinement, in this case, the initial policy corresponds to a symbol that is different from the desired ground truth, resulting in larger initial errors.

Two symbols were explored for evaluating COACH for policy reuse. First, a primitive for the letter "z" was used as an initial policy for the task of drawing a "2". The second was a "V" used for drawing an "U".

In Figure 13, the left-hand side shows the reference letter (blue) used for recording the demonstrations for the initial policy, as well as the final policy (red), which was subsequently obtained through the corrective process with COACH. The right-hand side shows the same final policy for the symbol "z" (red), which is used as the initial policy for learning the symbol "2", and its baseline (blue).

In this experiment, the user has to provide the corrective feedback during 10 episodes of the path execution. Figure 14 shows the evolution of the error through the episodes of correction during policy execution. Before the correction of the trajectories, the average error was 0.1717. Unlike in the policy refinement experiments, in this case, at the first episode, the users advised large changes of the policy, decreasing the initial error by about 90%. By the fourth episode, the corrections obtained a 99% error reduction.

Moreover, it is possible to observe that, by the third episode of corrections, the error was reduced to a level lower

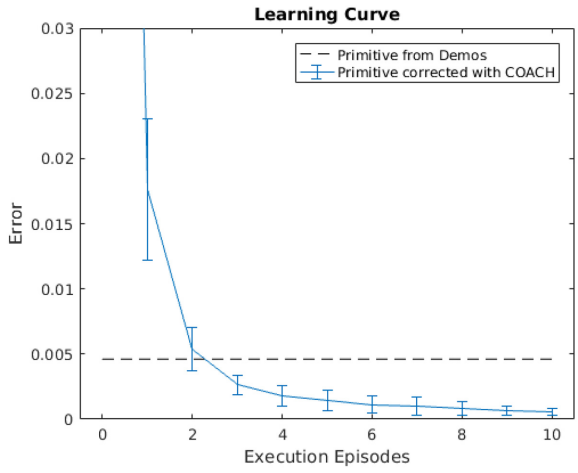


Fig. 14. Policy reuse; average error for learning symbols.

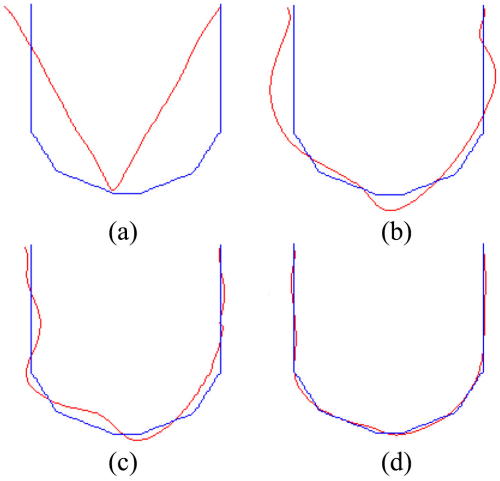


Fig. 15. Trajectory correction progress for policy reuse: learned primitive (red), symbol of reference (blue). (a) Policy without corrections, (b) after one correction episode, (c) after five episodes, (d) after ten episodes.

than the one obtained by a policy derived from datasets of five demonstrations; by the fifth episode, the percentage of error reduction was around 68%, also with respect to the policy obtained from demonstrations, which is a similar error obtained in the experiments of policy refinement. Figure 15 depicts the progress of the shape drawn for the symbol "U", where the initial policy is for drawing a "V".

6.2. Learning with simultaneous corrective advice and Policy Search

So far, trajectories were optimized purely by human feedback with COACH. In this section, we validate the combination of the Policy Search method with the human-guidance-based exploration using COACH in well-known problems in simulation and with a real robot.

In simulation, experiments were carried out using an arm with varying DoF in a reaching a via-point task. In a second set of experiments, the "ball in the cup" game was learned using a real robot. In both tasks, we compared the proposed interactive Policy Search strategy with a standard Policy Search algorithm, in terms of the convergence rate and final performance of the policies. Although the proposed hybrid method can be implemented with different Policy Search algorithms, in this experimental procedure both the standard and the hybrid Policy Search method are based on PI^2 (Theodorou et al., 2010).

6.2.1. Learning multi-DoF via-point movement tasks. The first set of experiments for the validation of the hybrid approach is carried out by replicating the experiments intended to compare Policy Search algorithms by Theodorou et al. (2010), and that has been repeated by Stulp and Sigaud (2012, 2013). The experiment involved learning robot arm reaching movements (similar to human reaching movements) with a total duration of 0.5 s. The task has the condition of reaching a specific via-point at $t = 0.3$ s, which is an approximation to hitting movements, because they require time–space synchronization.

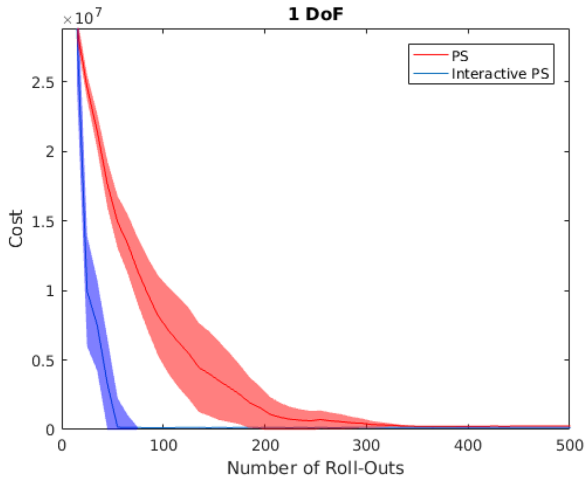
The learning task is evaluated in four different cases: first, with a one-dimensional moving point (one DoF); the next three cases are with planar arms of 2, 10, and 50 DoF, respectively. The policies are represented with DMPs, which compute the actions in the joint space for the multi-DoF tasks. The experiments were executed first with the original Policy Search algorithm PI^2, and followed by our hybrid approach, combining the Policy Search method and the COACH variation for policies defined in the joint and Cartesian space. Five users between 23 and 30 years old participated in these experiments; two participants were engineers while the others did not have a technical background. For every explored case, 20 runs of 500 roll-outs were executed for each of the algorithms. The obtained results are averaged and presented with their standard deviation.

One DoF via-point task. In this task, the initial position of the movement is $y_{t0} = 0$, and the DMP has the goal attractor $g = 1$ in order to finish the movement with $y_{t500\text{ms}} \approx 1$. The cost function is $r_t = 0$ for all time steps except in $t = 300$ms, as shown in equation (7), where G is the via-point set to $G = 0.25$

$$r_{300\text{ms}} = 10^8 (G - y_{t300\text{ms}})^2 \tag{7}$$

When participating in the learning process, the user observes the movement execution and advises the binary correction with a keyboard, similarly to the interaction in the experiments of learning to write letters, but only using two keys.

In Figure 16, the evolution of the cost function through the roll-outs execution is shown. The human feedback



Fig/ 16. Learning curve of the one degree of freedom (DoF) via-point movement task. Average and ± 1 standard deviation of 20 runs. PS: Policy Search.

supporting the Policy Search improvement makes a significant difference regarding the original Policy Search algorithm. The convergence time is reduced by one order of magnitude; the interactive Policy Search method is about 83% faster than the conventional Policy Search method. Moreover, it is possible to see that the variance of the cost function is decreased with the human guidance.

Multi-DoF via-point tasks. In these cases of simulated planar arms, the initial position is $a = 0$ for all the joint angles; it sets a robot pose that is a straight line parallel to the horizontal axis. The goal attractor x_{goal} results in a semicircular configuration, as shown in Figure 17, where the end effector of a 10 DoF arm touches the y axis. The end effector is moving in two-dimensional space, and has to pass through the via-point $G = (0.5, 0.5)$.

Figure 17(a) shows the trajectory of the arm with the initial policy of the learning process. In Figure 17(b), the end effector is already passing through the via-point.

The objective of the cost function is to reduce the distance between the end effector and the via-point at $t = 300$ ms. Additionally, it also tries to reduce joints accelerations, giving more priority to the joints that are closer to the "shoulder" of the arm, with D the number of DoF of the robotic arm, as

$$r_t = 10^8 \delta(t - 300\text{ms}) \cdot ((x_t - 0.5)^2 + (y_t - 0.5)^2) + \frac{\sum_{d=1}^{D} (D + 1 - d)(\ddot{a}_{d,t})^2}{\sum_{d=1}^{D} (D + 1 - d)} \quad (8)$$

During the interactive learning process, the user advises the corrections with binary corrections in both axes, as was done for correcting the letters in the previous subsection. In previous works, these experiments have been carried out for learning policies in the joint space domain; nevertheless,

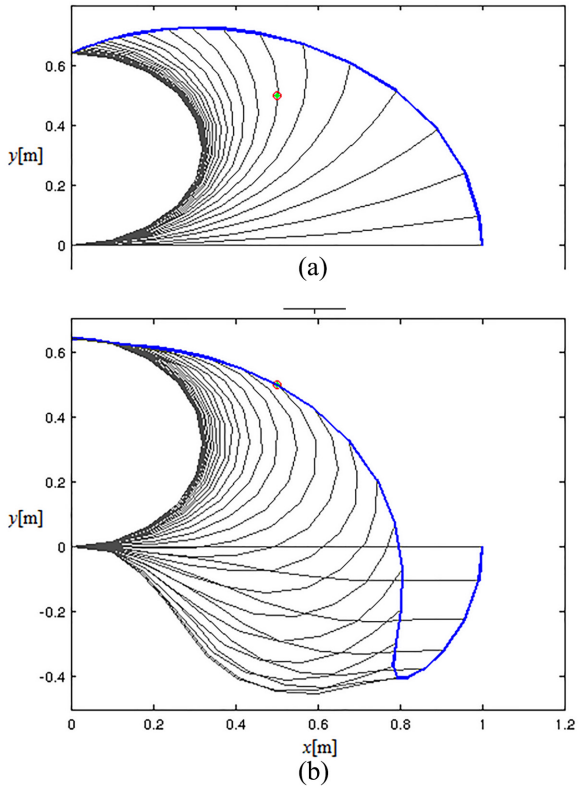


Fig. 17. "Stroboscopic" visualization of the 10 DoF planar robot arm movement: (a) simply toward the goal, (b) through the via-point (green/red dot).

in this paper we approach the problem in both the Cartesian and the joint domains.

The learning curves in Figures 18 to 20 show the improvement achieved when the Policy Search method takes the human corrective feedback into consideration. The general trend shown in the curves is that the hybrid agents converge faster than the standard Policy Search method. In the cases of 10 and 50 DoF, the standard Policy Search method learns faster in the Cartesian domain than in the joint space, owing to the smaller search space. In contrast, the interactive Policy Search method learns faster when learning policies in joint space compared with when learning policies represented in the end-effector domain.

Since the original problem is only explored with policies in the joint space, and also because the best obtained policies are in that domain, the rest of the analysis is only focused on the comparison between standard and interactive Policy Search methods for learning policies in joint space.

For the task with the two DoF arm the interactive Policy Search method decreases the initial cost by 95% within the first 50 roll-outs, and then maintains a slight rate of improvement, reaching 97.9% by the 500th trial. In contrast, the conventional Policy Search method attains the 95% of cost reduction approximately after 210 trials, i.e., it is four times slower than the interactive Policy Search

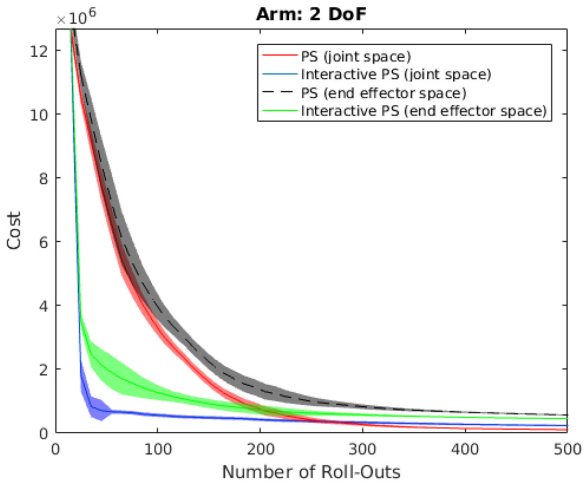


Fig. 18. Learning curve of the two degrees of freedom (DoF) via-point movement task. Average and ± 1 standard deviation of 20 runs.
PS: Policy Search.

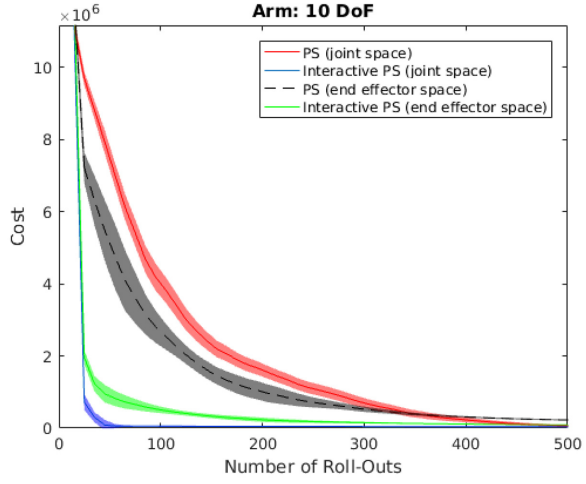


Fig. 19. Learning curve of the 10 degrees of freedom (DoF) via-point movement task. Average and ± 1 standard deviation of 20 runs.
PS: Policy Search.

method. However, the conventional Policy Search method keeps the error reduction, and outperforms the performance of the interactive Policy Search method after 280 episodes, reaching a total reduction of 99.2% with 500 episodes.

In the experiments with the 10 DoF arm, results are similar, but with a greater difference between the cost of both algorithms. The 95% reduction is obtained during the first 30 trials with the interactive Policy Search method, whereas the conventional Policy Search method is 11 times slower for achieving that performance and after 500 roll-outs reaches the curve of the interactive Policy Search method.

For the last case, with 50 DoF, again the difference is increased drastically, as 500 roll-outs are not enough for

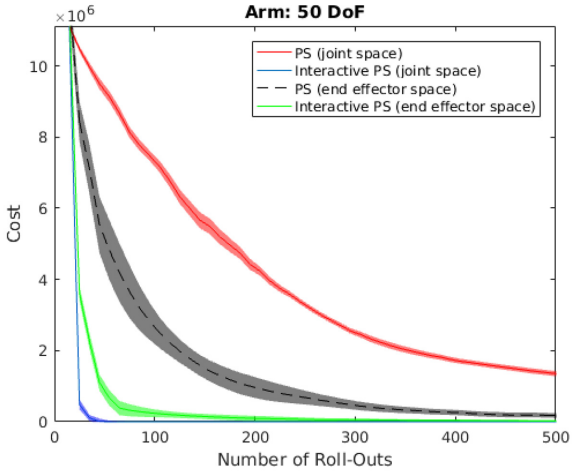


Fig. 20. Learning curve of the 50 degrees of freedom (DoF) via-point movement task. Average and ± 1 standard deviation of 20 runs.
PS: Policy Search.

the Policy Search agent to converge in this problem. Then, at the end of the learning process, the Policy Search agent only achieved a 86.9% decrease of the initial cost. Conversely, the interactive Policy Search method achieved a 95% reduction by approximately 25 episodes and converged completely after the 60th episode.

In the previously presented results, the convergence of the Policy Search method is affected when the number of DoF is increased, owing to the curse of dimensionality. But the convergence of the proposed interactive Policy Search method shows a counterintuitive trend after the increase of DoFs. Indeed, the fastest convergence obtained using human feedback is in the case of the arm with 50 DoF. The reason behind this effect has to do with the correspondence problem between the corrective advice in the effector domain, which is mapped to the joint space, where the policy is defined. When the human teacher advises a correction to the two DoF arm, in several cases the solution found by the inverse kinematics model could be a joint configuration very different from the previous one, or it simply could not find a proper solution that matched the end-effector position correction.

These problems cause policy updates that do not match the user's intention; therefore, the teacher's correction might harm the policy from time to time. Nevertheless, when the task has more DoF, this problem diminishes. The more redundant the arm is, the easier it is to shape the end-effector trajectory with corrective advice. Although, for the case of the two DoF arm, the corrective advice does not work perfectly, the Policy Search method still benefits strongly from the human guidance and reduces the convergence time by 76%, compared with the conventional Policy Search method.

As in the experiments of learning to write symbols, for this case, the amount of corrections advised during the

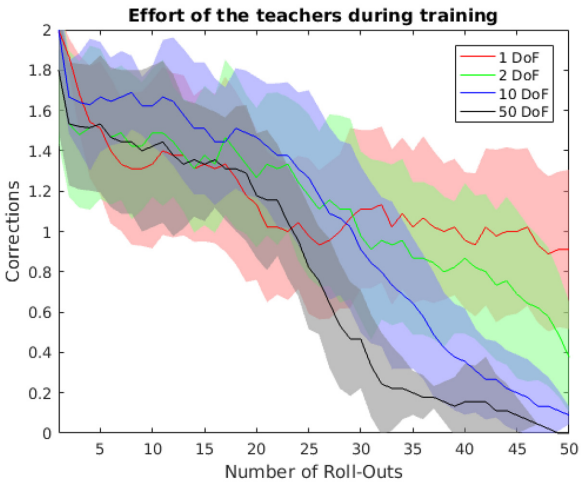


Fig. 21. Average of the corrections advised by the teachers for correcting the multi-DoF via-point movements.
DoF: degrees of freedom.

episodes is presented in Figure 21. The curves show that the users advised almost two corrections per episode at the beginning, which is extremely less information than that needed to demonstrate the complete trajectory, and also easier for non-expert users, who probably cannot successfully demonstrate accurate movements at this speed. There is a correlation between the reduction of the corrections and the reduction of the cost in the learning curves; this is because the users decrease the frequency of the corrections when the performance is better, for instance with the arm of 50 DoF, the learning curves in Figure 20 are the fastest, whereas in Figure 21 it is possible to see that it needs the least amount of corrections. These results show the potential of the proposed method for learning policies that need to synchronize movements in time and space based on reinforcement learning, as it requires very few of this kind of binary coarse correction.

6.2.2. Ball-in-a-cup game. The *ball-in-a-cup* game is a challenging children's game that requires accurate skills in a relatively fast movement. The game uses a toy composed of a ball attached to a cup with a string. The cup is held with the hand of the player, or, in this case, attached to the end effector of the robot. Initially, the ball hangs steady below the cup, and the arm has to move the cup fast enough to launch the ball high in the air to catch it during the landing.

A reward function that represents the task objective would be one that punishes a failed trial and rewards when the ball falls into the cup. However, such a function is not informative for efficient (or even feasible) learning in a reinforcement learning setting. This problem was approached by Kober and Peters (2009) with a more complex function of the form

$$r_t = \begin{cases} \exp\left(-\alpha(x_c - x_b)^2 - \alpha(y_c - y_b)^2\right) & \text{if } t = t_c \\ 0 & \text{otherwise} \end{cases} \quad (9)$$

using the Policy Search algorithm PoWER on a real robotic arm Barrett WAM.

In this reward function, the ball and cup positions are $[x_b, y_b, z_b]$ and $[x_c, y_c, z_c]$, respectively. The time $t = t_c$ is the moment when the ball passes the rim of the cup with downward direction; for all the other time steps $t \neq t_c$, the function is $r_t = 0$. This reward function was used by the Policy Search method to improve an initial policy obtained from a human demonstration.

In this work, we validate the proposed interactive Policy Search method with this problem using a seven DoF KUKA lightweight arm and an OptiTrack system, which tracks the position of the ball and the cup, to compute the reward function. Nevertheless, there are two important differences with respect to the work of Kober and Peters (2009). First, here the policy computes the trajectory of the end effector instead of computing actions in the joint space; secondly, we consider not only how to improve policies learned from human demonstrations, but also how to learn the policies from scratch; therefore, the reward function of equation (9) is complemented in order to make it more informative.

When the arm tosses the ball with a height lower than the cup, equation (9) is completely uninformative, as it always results in a zero. To lead the policy to a behavior wherein equation (9) is applicable, we propose to complement this function with a term that rewards the height obtained in those cases.

Then, when the ball does not reach the height of the cup

$$r_{t_h} = \frac{z_b - z_c}{l_s + l_c} \quad (10)$$

is applied, where l_c and l_s denote the length of the cup and the string, respectively, and $t = t_h$ is the moment when the ball reaches the maximum height. The sum of l_c and l_s is the distance between the ball and the rim of the cup when the ball is hanging motionless.

With this extension, the "ball-in-a-cup" task can be considered a composition of the sub-task "swinging the ball" with the objective of tossing the ball higher than the cup, followed by the sub-task "catching the ball', which aims to move the arm for intercepting the ball with the cup. For the computations of the learning algorithms, the reward function is transformed into a cost function by multiplying it by -1.

Unlike previous works that use DMPs for this problem, we opted to represent the policy with the ProMP form of equation (4), as in the experiments of Section 6.1, since the convergence to goal attractors is not a necessary property for this task. Again, PI^2 is the base Policy Search algorithm for these experiments.

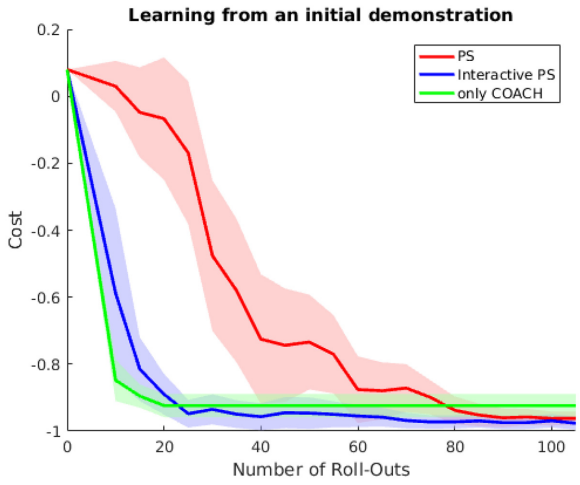


Fig. 22. Convergence curves for learning the "ball-in-a-cup" game with an initial demonstration. Average and ± 1 standard deviation of 20 runs.
PS, Policy Search.

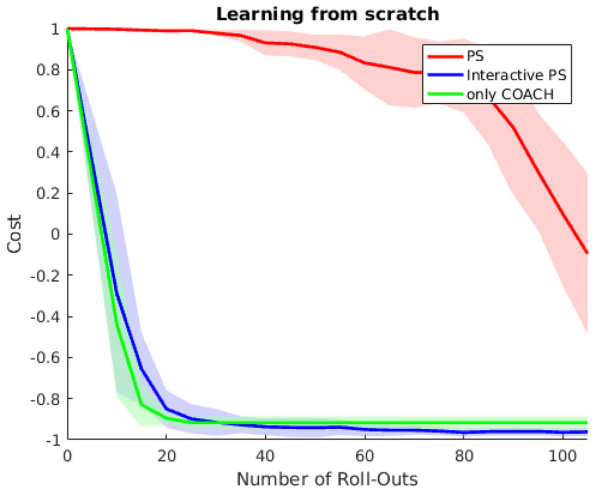


Fig. 23. Convergence curves for learning "ball-in-a-cup" game from scratch. Average and ± 1 standard deviation of 20 runs.
PS, Policy Search.

In the experimental procedure, for the learning processes, cups of two different sizes were used to change the difficulty of the task. The big cup (approximately twice the diameter of the ball) was attached to the robot arm via a stick, see Figure 1. The big cup also contained the small cup (diameter approximately 1.3 times the ball's diameter). During learning processes, the human teacher sat in front of the robot,with a perspective similar to that in Figure 1.

Most of the validation experiments were carried out with the big cup. The standard Policy Search method was compared with learning with human feedback in the interactive Policy Search approach, along with the pure COACH method, executing 10 runs for each approach. In this experiment, two participants performed as teachers; one was an author of this paper, and the other was a person without technical experience. Since the learned policy computes actions in the end-effector domain, the two interactive methods are based on the complete COACH approach for training motor primitives in the Cartesian space. A keyboard was used to advise the corrections by the user.

The first set of experiments started with a policy derived from a kinesthetic demonstration and using the big cup. The learning curves presented in Figure 22 show the large difference between learning with the Policy Search and interactive methods. The Policy Search method achieves policies that catch the ball with the cup at around the 60th trial, and keep improving until convergence after 90 trials. In contrast, with COACH and the interactive Policy Search method, the task is achieved after 10 and 15 episodes, respectively. With COACH, the improvement stops very quickly, because on observing a successful policy, a human use reduces the effort for enhancing it, either because the need for improvement is not particularly evident or improvement is not necessary. For human teachers, it is hard to infer corrections when suboptimal policies are close to the optimal.

Conversely, at the beginning, the interactive Policy Search method is slightly slower than the pure COACH method, owing to the influence of some of the first roll-outs in the update process. However, with the hybrid method, the improvement continues until the 70th episode, reaching the best average performance. This improvement results from both sources of feedback, especially from the reward function.

The learning curve of the Policy Search method reaches and outperforms the cost obtained with COACH within 75–80 trials, i.e., it needs four times more trials. However, within 100 trials, Policy Search does not attain the performance obtained by interactive Policy Search.

For a second set of experiments, a more challenging scenario was used to test the learning algorithms, wherein a previous demonstration was not given by the human teacher (learning from scratch). Therefore, a policy that did not request any movement was set at the beginning of the learning process.

Since the arm was not able to toss the ball to the necessary height in one shot, the robot needed to learn the "swinging the ball" sub-task, so that the ball would oscillate like a pendulum to obtain enough momentum. In that first part of the learning process, the term in equation (10) of the reward function plays an important role. When the policy evolves, maximizing equation (10), it continues learning to catch the ball with the cup using equation (9).

The results of these experiments are shown in Figure 23, which shows that these interactive methods, based on vague corrective advice, are more robust to the initial policy than the standard Policy Search method. Both learning curves of the methods based on corrective advice are basically the same as the experiments for learning with an initial

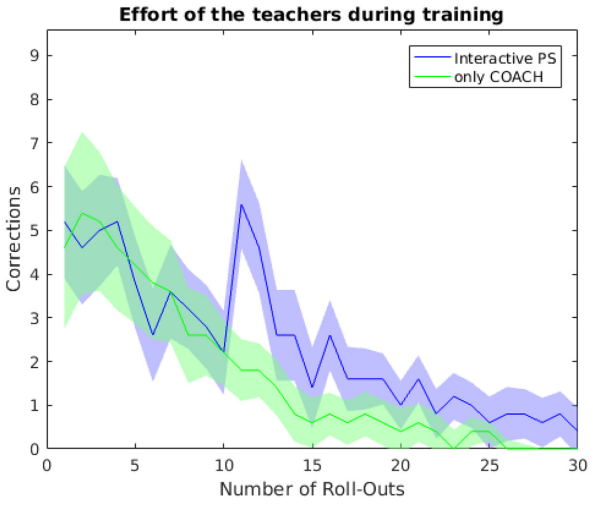


Fig. 24. Average of the corrections advised by the teachers for correcting the "ball-in-a-cup" movements.
PS, Policy Search.

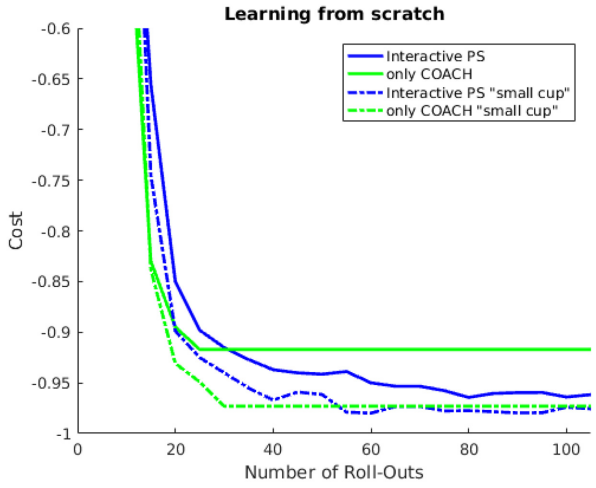


Fig. 25. Comparison of learning curves for the "ball-in-a-cup" task, in the scenario of learning from scratch using the big and small cups.
PS, Policy Search.

demonstration, but delayed by approximately five trials, owing to the episodes intended to learn to swing the ball.

The convergence of Policy Search method is very sensitive to the initial policy. In this scenario, the Policy Search method takes about 170 episodes to attain successful policies. The very beginning of the learning process is very slow because the random movements tend to diminish the effect of the previous ones, even considering that the algorithm implementations of this work use state-dependent exploration, as in Kober and Peters (2009) and Theodorou et al. (2010) for avoiding high-frequency actions.

Finally, the interactive Policy Search method keeps optimizing the policy when the improvement is not evident for the human teachers, outperforming the outcomes obtained with only COACH after 30 trials.

Figure 24 shows the number of corrections advised during the roll-outs. Like the conclusion obtained from the experiments of the multi-DoF via-point movements, the faster the cost reduction, the fewer advised corrections. In this case, using only COACH requires less corrections than the combined method; this is because the Policy Search updating process makes the progress slightly slower, as shown in Figures 22 and 23.

The set-up with the small cup was used to compare the interactive algorithms while learning from scratch. In Figure 25, the means of the learning curves based on human feedback presented in Figure 23 were taken as a reference to be compared with the experiments of learning using the small cup.

Since the cost function is the same, and basically takes into account the distance of the ball to the center of the cup, the size of the cup would not affect the cost evolution with an algorithm based only on the computed reward for updating the policy, like the Policy Search algorithm. But in the cases of the methods with human feedback, it is possible to see that the cost is decreased faster in the set-up with the small cup.

The previous counterintuitive observation is due to problems with the human perception, because for a teacher it is hard to know whether the ball is falling into the cup exactly through its center or not. The users may not be able to estimate depth accurately, so when teachers are in front of the robot, it is harder to make the ball to pass through the x and y center of the big cup. With the smaller cup, the rim of the cup itself is a visual aid. Then, when the ball hits the rim, the user infers the correction based on that visual information. Therefore, when the policy makes the ball fall into the cup without touching the rim, it is already crossing its center or very close to it.

With the small cup, the users could track the progress of the policy better, so Figure 25 shows that users were engaged with the learning process during more trials, e.g., when learning with only COACH and the big cup, the policy improvement stopped at roll-out 25, whereas with the small cup they kept correcting during five more roll-outs.

In the previous experiments, the last part of the improvement using the interactive Policy Search method is mostly based on the reward function. However, this last experiment with the small cup shows that in cases wherein the human perception is enough to obtain insights about how good a policy is when it is close to the optimum (rarely), with only COACH is possible to achieve performances like the obtained with the interactive Policy Search method. Extension 4 shows the learning process of the task with the proposed learning method.[4]

7. Conclusions

In this paper, we have proposed the use of human corrective feedback within the framework of Policy Search

methods for learning movement primitives. First, the application of pure corrective advice for adapting parametrized trajectories during time execution was presented as a simple extension of the COACH framework. Secondly, this extension was integrated in the exploration stage of standard Policy Search algorithms in order to combine both sources of improvement: (1) random exploration and (2) human corrections.

Schemes based on pure human corrective advice showed that this type of relative correction with vague binary signals provide human teachers with the capability of modifying trajectories while a robot executes it. The experiment of writing symbols showed that users can obtain very good shapes for the symbols with corrective feedback. The users obtained better policies using corrective advice than they did solely from demonstrations, which means that the application of corrective advice renders it less necessary to have users with a high level of expertise in the task and in using human–robot interfaces.

The validation of the proposed interactive Policy Search method showed outstanding results in two well-known benchmark problems with simulated and real robots. The learning curves showed that the proposed method speeds up the convergence of Policy Search by between 4 and more than 40 times. Human feedback is extremely powerful in accelerating the learning process at the beginning, whereas the cost function has an important influence for performing fine-tuning when suboptimal policies have a good performance, but the users' perception is not able to determine how more improvement can be obtained through corrections, e.g., when the policy already accomplishes the task but the energy used can be further reduced.

The validation of the proposed hybrid learning scheme showed that it is possible to learn complex skills, such as those required to solve the"ball-in-a-cup" task, without previous demonstrations. This method allows learning processes to be started from scratch, i.e., initial static policies, which incrementally receive the user's corrective advice. Little by little, the human teachers guide the robot to policies that satisfy their understanding about the fulfillment of the task. Also, the results show that it is possible to learn this skill based only on human corrections.

Moreover, the proposed strategy to cope with the correspondence problem—matching the human feedback given in the task domain with the policy in the joint space—has shown that the method scales to high-dimensionality problems; in fact, in problems using multi-DoF planar arms, the best results obtained are for the arm of most DoF.

The corrective feedback used in COACH is limited to applications in which the teacher can observe the world, evaluate, and advise corrections according to his or her understanding of the task, and the dynamics of the environment. For problems with fast and complex transitions in high-dimensional action spaces, the user would not always be able to give appropriate advice, e.g., when learning to control a drone; the learning curve of our proposed interactive Policy Search would then be similar to one of pure reinforcement learning, since few feedback signals would be given by the teacher. Therefore, there is still a need to extend these methods to approaches that transform the problems to scenarios wherein the teacher can advise, for instance, providing feedback in offline playback, or simultaneous policy and model learning for carrying out part of the training in simulation.

More future work is intended to extend this learning scheme for policies parameterized with deep neural networks, so the advantages of the method proposed in this work can be obtained for learning end-to-end policies. Also, further research will consider how to learn reward functions from human corrective feedback, in order to apply reinforcement learning in situations wherein there is no reward function available, and it is not possible to record expert demonstrations.

Acknowledgements

The authors would like to thank Rui Silva, Manuela Veloso at Carnegie Mellon University, and Dorothea Koert and Marco Ewerton at Technische Universität Darmstadt for their constructive and valuable discussions during the development of this work.

Funding

The author(s) disclosed receipt of the following financial support for the research, authorship, and/or publication of this article: This work was partially funded by FONDECYT project [1161500], CONICYT project [AFB180004], CONICYTPCHA/Doctorado Nacional/2015-21151488, the European Community's Seventh Framework Programme [FP7-ICT-2013-10] under grant agreement 610878 (3rdHand), and by the Japanese New Energy and Industrial Technology Development Organization (NEDO).

Notes

1. https://youtu.be/ptslNZdum2s?t=3
2. https://youtu.be/ptslNZdum2s?t=24
3. https://youtu.be/ptslNZdum2s?t=67
4. https://youtu.be/ptslNZdum2s?t=129

References

Akrour R, Schoenauer M and Sebag M (2011) Preference-based policy learning. In: Gunopulos D, Hofmann T, Malerba D, et al. (eds) *Machine Learning and Knowledge Discovery in Databases. ECML PKDD 2011*. Berlin: Springer, pp. 12–27.

Akrour R, Schoenauer M, Sebag M, et al. (2014) Programming by feedback. *Proceedings of Machine Learning Research* 32(2): 1503–1511.

Amershi S, Cakmak M, Knox WB, et al. (2014) Power to the people: The role of humans in interactive machine learning. *AI Magazine* 35(4): 105–120.

Argall BD, Browning B and Veloso M (2008) Learning robot motion control with demonstration and advice-operators. In: *IEEE/RSJ international conference on intelligent robots and systems (IROS)*, Nice, France, 22–26 September 2008, pp. 399–404. Piscataway, NJ: IEEE.

Argall BD, Chernova S, Veloso M, et al. (2009) A survey of robot learning from demonstration. *Robotics and Autonomous Systems* 57(5): 469–483.

Argall BD, Sauser EL and Billard A (2011) *Tactile Guidance for Policy Adaptation*, vol. 2. Boston, MA: Now Publishers Inc.

Breazeal C and Scassellati B (2002) Robots that imitate humans. *Trends in Cognitive Sciences* 6(11): 481–487.

Canal G, Alenyà G and Torras C (2016) Personalization framework for adaptive robotic feeding assistance. In: Agah A, Cabibihan JJ, Howard A, et al. (eds) *International Conference on Social Robotics. Social Robotics. ICSR 2016*. Cham: Springer, pp. 22–31.

Celemin C and Ruiz-del Solar J (2018) An interactive framework for learning continuous actions policies based on corrective feedback. *Journal of Intelligent & Robotic Systems* 95(1): 77–97.

Chernova S and Thomaz AL (2014) Robot learning from human teachers. *Synthesis Lectures on Artificial Intelligence and Machine Learning* 8(3): 1–121.

Christiano P, Leike J, Brown TB, et al. (2017) Deep reinforcement learning from human preferences. arXiv arXiv:1706.03741.

Deisenroth MP, Neumann G and Peters J (2013) A survey on Policy Search for robotics. *Foundations and Trends in Robotics* 2(1–2): 1–142.

Ewerton M, Maeda G, Kollegger G, et al. (2016) Incremental imitation learning of context-dependent motor skills. In: *IEEE-RAS 16th international conference on humanoid robots (humanoids)*, Cancun, Mexico, 15–17 November 2016, pp. 351–358. Piscataway, NJ: IEEE.

Hansen N and Ostermeier A (2001) Completely derandomized self-adaptation in evolution strategies. *Evolutionary Computation* 9(2): 159–195.

Ijspeert AJ, Nakanishi J and Schaal S (2002) Movement imitation with nonlinear dynamical systems in humanoid robots. In: *IEEE international conference on robotics and automation (ICRA)*, Washington, DC, USA, 11–15 May 2002, vol. 2, pp. 1398–1403. Piscataway, NJ: IEEE.

Jain A, Wojcik B, Joachims T, et al. (2013) Learning trajectory preferences for manipulators via iterative improvement. *Advances in Neural Information Processing Systems (NIPS)* 26: 575–583.

Khansari-Zadeh SM and Billard A (2011) Learning stable nonlinear dynamical systems with Gaussian mixture models. *IEEE Transactions on Robotics* 27(5): 943–957.

Knox WB and Stone P (2009) Interactively shaping agents via human reinforcement: The tamer framework. In: *Fifth international conference on knowledge capture*, Redondo Beach, CA, USA, 1–4 September 2009, pp. 9–16. New York, NY: ACM.

Kober J and Peters J (2009) Policy search for motor primitives in robotics. *Advances in Neural Information Processing Systems (NIPS)*. 21: 849–856.

Kober J, Bagnell JA and Peters J (2013) Reinforcement learning in robotics: A survey. *The International Journal of Robotics Research* 32(11): 1238–1274.

Kober J, Wilhelm A, Oztop E, et al. (2012) Reinforcement learning to adjust parametrized motor primitives to new situations. *Autonomous Robots* 33(4): 361–379.

Kormushev P, Calinon S and Caldwell DG (2010) Robot motor skill coordination with em-based reinforcement learning. In: *IEEE/RSJ international conference on intelligent robots and systems (IROS)*, Taipei, Taiwan, 18–22 October 2010, pp. 3232–3237. Piscataway, NJ: IEEE.

Maeda G, Ewerton M, Koert D, et al. (2016) Acquiring and generalizing the embodiment mapping from human observations to robot skills. *IEEE Robotics and Automation Letters* 1(2): 784–791.

Meriçli Ç, Veloso M and Akin HL (2011) Task refinement for autonomous robots using complementary corrective human feedback. *International Journal of Advanced Robotic Systems* 8(2): 16.

Metzen JH, Fabisch A, Senger L, et al. (2014) Towards learning of generic skills for robotic manipulation. *KI-Künstliche Intelligenz* 28(1): 15–20.

Paraschos A, Daniel C, Peters J, et al. (2013) Probabilistic movement primitives. *Advances in Neural Information Processing Systems (NIPS)* 26: 2616–2624.

Peters J and Schaal S (2008) Reinforcement learning of motor skills with policy gradients. *Neural Networks* 21(4): 682–697.

Peters J, Mülling K and Altun Y (2010) Relative entropy Policy Search. In: *Conference on artificial intelligence (AAAI)*. Atlanta, GA, USA, 11–15 – July 2010, pp. 1607–1612. Menlo Park, CA: AAAI Press.

Pongas D, Billard A and Schaal S (2005) Rapid synchronization and accurate phase-locking of rhythmic motor primitives. In: *IEEE/RSJ international conference on intelligent robots and systems (IROS)*, Edmonton, Canada, 2–6 August 2005, pp. 2911–2916. Piscataway, NJ: IEEE.

Schaal S, Ijspeert AJ and Billard A (2003) Computational approaches to motor learning by imitation. *Philosophical Transactions of the Royal Society of London B: Biological Sciences* 358(1431): 537–547.

Schroecker Y, Ben Amor H and Thomaz AL (2016) Directing Policy Search with interactively taught via-points. In: *International conference on autonomous agents & multiagent systems*, Singapore, 9–13 May 2016, pp. 1052–1059. Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems.

Stulp F and Sigaud O (2012) Path integral policy improvement with covariance matrix adaptation. arXivarXiv: 1206.4621.

Stulp F and Sigaud O (2013) Robot skill learning: From reinforcement learning to evolution strategies. *Paladyn, Journal of Behavioral Robotics* 4(1): 49–61.

Suay HB, Toris R and Chernova S (2012) A practical comparison of three robot learning from demonstration algorithm. *International Journal of Social Robotics* 4(4): 319–330.

Theodorou E, Buchli J and Schaal S (2010) A generalized path integral control approach to reinforcement learning. *Journal of Machine Learning Research* 11: 3137–3181.

Thomaz AL and Breazeal C (2006) Adding guidance to interactive reinforcement learning. In: *Proceedings of the Twentieth Conference on Artificial Intelligence (AAAI)*.

Thomaz AL and Breazeal C (2007) Asymmetric interpretations of positive and negative human feedback for a social learning agent. In: *16th IEEE international symposium on robot and human interactive communication (RO-MAN)*, Jeju, South Korea, 26–29 August 2007, pp. 720–725. Piscataway, NJ: IEEE.

Thomaz AL, Hoffman G and Breazeal C (2006) Reinforcement learning with human teachers: Understanding how people want to teach robots. In: *15th IEEE international symposium on robot and human interactive communication (RO-MAN)*, pp. 352–357. Piscataway, NJ: IEEE.

Appendix: Index to multimedia extensions

Archives of IJRR multimedia extensions published prior to 2014 can be found at http://www.ijrr.org, after 2014 all videos are available on the IJRR YouTube channel at http://www.youtube.com/user/ijrrmultimedia

Table of multimedia extensions.

Extension	Media type	Description
1	Video	User interface for recording demonstrations
2	Video	Learning to write characters with a simulated robot arm
3	Video	Learning to write characters with a real robot arm
4	Video	The “ball-in-a-cup” game