Advanced Robotics, 2013
Vol. 27, No. 13, 1023–1031, http://dx.doi.org/10.1080/01691864.2013.814211

FULL PAPER

Efficient sensorimotor learning from multiple demonstrations

Bojan Nemec*, Rok Vuga and Aleš Ude

Humanoid and cognitive robotics lab, Department of automatics, biocybernetics and robotics, Jožef Stefan Institute, Jamova 39, 1000 Ljubljana, Slovenia

(*Received 25 December 2012; accepted 26 April 2013*)

In this paper, we present a new approach to the problem of learning motor primitives, which combines ideas from statistical generalization and error learning. The learning procedure is formulated in two stages. The first stage is based on the generalization of previously trained movements associated with a specific task configuration, which results in a first approximation of a suitable control policy in a new situation. The second stage applies learning in the subspace defined by the previously acquired training data, which results in a learning problem in constrained domain. We show that reinforcement learning in constrained domain can be interpreted as an error-learning algorithm. Furthermore, we propose modifications to speed up the learning process. The proposed approach was tested both in simulation and experimentally on two challenging tasks: learning of matchbox flip-up and pouring.

Keywords: robot learning; trajectory generation; error learning

1. Introduction

One of still unresolved issues in the contemporary robotics is how to create a fully autonomous robot, which makes decisions based solely on its sensors, feedback and using previous experience. In order to achieve this goal, efficient and robust learning algorithms are needed. Among the most promising frameworks to bring traditional robotics towards true autonomy is the reinforcement learning (RL). One of the problems in RL of robot actions is a potentially huge search space. Robot actions are usually encoded using parameterized motor primitives, where the number of parameters required to encode the robot skill is relatively high. This makes RL extremely challenging. Recently, new algorithms such as Policy Improvement with Path Integrals (**PI**^{2}) [1] and Policy Learning by Weighting Exploration with the Returns (PoWER) [2] were developed to handle efficiently learning in high dimensions.

However, despite these efforts, learning capabilities of robots still cannot be compared to those of humans. Humans can quickly adapt to new situations by generalization. In contrast, robots often have to relearn whole trajectories, even when good initial policy parameters are provided. It turns out that the initial guess of search direction in the learning process is in most cases more important than the initial guess of the parameters itself. Many approaches dealing with the problem of how to make learning more efficient rely on reducing the number of learned parameters. It often turns out that only a limited number of parameters are relevant for a specific task. For example, Scholz and Schöher [3] studied the stand-up task in humans and found out that the center of mass is among the most relevant parameters. Kormushev et al. [4] dealt with archery skill using a humanoid robot and suggested an algorithm, where the parameter updates are formed as a linear combination of parameters reweighted according to the reward in the previous trial. Grollman and Billard [5] proposed a learning method, where the search space is constrained to lie between two unsuccessful demonstrations. For the underlying representation, they used Gaussian Mixture Models (GMMs). Recently, another approach was suggested by Kober et al. [6], where the sensorimotor policy was improved by adapting a small set of global parameters, called meta-parameters. Similar idea was evaluated in [7], where we combined ideas of statistical generalization and RL in order to achieve the same goal. The key idea was to limit the potentially huge search space of the parameterized policy by using previous experience and to generalize to new policies from similar cases. Moreover, generalization from similar cases provides good initial guess of the search directions. Instead of tuning of all policy parameters, the proposed method learns the appropriate query regarding the desired goal from a simplified and sparse statistical model.

In this paper, we show that RL in constrained domain can be represented as an error-learning algorithm.[8] In this case, the learning problem turns into the problem of finding the zero of a unimodal function, which enablesus to use

*Corresponding author. Email: bojan.nemec@ijs.si the well-developed theory of the line-search algorithms.[9] This way we minimize the number of trials required to learn the desired sensorimotor policy. We evaluate these ideas on two challenging tasks: learning to flip up a matchbox and pouring. In the first case, the robot has to learn how to hit the matchbox lying on a table in such a way that it flips upright. In the second case, the robot has to learn how to pour an equal quantity of liquid into a glass from bottles containing different volumes. The paper is organized in six chapters. In the second chapter, we present a statistical method for generalization of actions from previous examples, where we used Dynamic Motion Primitives (DMPs) for the underlying trajectory representation. In the third chapter, we introduce DMP parameters learning in constrained domain using the policy gradient RL. Error learning is evaluated in the fourth chapter. The simulation and experimental results are outlined in the fifth chapter. Final remarks and conclusions are given in the sixth chapter.

2. Action generalization from previous experiences

In this section we present the basic procedure for the generalization of actions to new situations that are not available in the database of movements. First we have to choose the appropriate action representation. For this purpose, we use dynamic movement primitives (DMPs) developed in [10]. With this representation, every degree of freedom is described by its own dynamic system, but with a common phase to synchronize them. In the case of point-to-point (discrete) movements, the trajectory of each robot's degree of freedom y (given either in joint or in task space) is described by the following system of nonlinear differential equations

$$\tau\dot{z} = \alpha_z(\beta_z(g - y) - z) + f(x), \tag{1}$$

$$\tau\dot{y} = z, \tag{2}$$

$$\tau\dot{x} = -\alpha_x x, \tag{3}$$

where x is the phase variable and z is an auxiliary variable. α_x, α_z, β_z, and τ are constants that need to be specified in such a way that the system converges to the unique equilibrium point $(z, y, x) = (0, g, 0)$. The nonlinear term f contains free parameters that enable the robot to follow any smooth point-to-point trajectory from the initial position y_0 to the final configuration g

$$f(x) = \frac{\sum_{k=1}^{N} w_k \Psi_k(x)}{\sum_{k=1}^{N} \Psi_k(x)} x,$$
$$\Psi_k(x) = \exp\left(-h_k\,(x - c_k)^2\right). \tag{4}$$

Here, c_k are the centers of radial basis functions distributed along the trajectory and $h_k > 0$ their widths. Weights w_k and the goal position g are computed in such a way that the DMP encodes the desired trajectory.

Lets assume that we have a set of example trajectories together with the parameters characterizing the task

$$\mathcal{Z} = \Big\{ y_{\mathrm{d}}^k(t_{k,j}),\, \dot{y}_{\mathrm{d}}^k(t_{k,j}),\, \ddot{y}_{\mathrm{d}}^k(t_{k,j});\, \mathbf{q}_k \,|\quad k = 1, \ldots, M,$$
$$j = 1, \ldots, T_k \Big\}, \tag{5}$$

where $y_d^k(t_j)$, $\dot{y}_d^k(t_j)$, and $\ddot{y}_d^k(t_j)$ are the measured positions, velocities, and accelerations on trajectory k, repsectively; M is the number of examples; and T_k is the number of sampling points on each trajectory. Indexing of the degrees of freedom is omitted from Equation (2) for clarity. $\mathbf{q}_k \in \mathbb{R}^n$ are the parameters describing the task in a given example situation. The trajectories can be specified in either joint or task space. The issue is how to generate a DMP specifying a movement for every new query point $\mathbf{q}$, which in general will not be one of the example queries $\mathbf{q}_k$.

To generalize the example movements to new situations, we need to learn a function

$$\mathbf{G}(\mathbf{q};\, \mathcal{Z}) \longmapsto [\mathbf{w}^T, \tau, g]^T = \boldsymbol{\theta}. \tag{6}$$

In general, the functional relationship between $\mathbf{q}$ and $[\mathbf{w}^T, \tau, g]^T$ given in a set of examples $\mathcal{Z}$ is unknown. Note that $\mathbf{G}(\mathbf{q};\, \mathcal{Z})$ becomes a function only by constraining the generalized trajectories to be as similar as possible to the example trajectories. For example, there are many different ways of how to throw the ball into the basket The relationship between the basket positions (query point) and DMP parameters describing the robot motion becomes a function only by requiring that the generalized movements are similar to the example movements. In most cases, it is difficult to find a global model that provides a good approximation for the function $\mathbf{G}(\mathbf{q};\, \mathcal{Z})$. We therefore avoid global model identification and rather apply regression techniques to generalize the movements. Due to significantly different sizes of data-sets involved in the calculation of parameters $\mathbf{w}$ on the one hand, and g and τ on the other hand, different methods can be utilized to estimate them. In particular, in [11] we applied locally weighted regression for the estimation of the shape parameters and Gaussian process regression [12] to estimate g and τ. More details about this work can be found in [11]. The important point for this paper is that a functional relationship between the query points and the DMP can be learned from example movements. The accuracy of the generalized motion depends on the nature of the problem and the number and density of the query points. If there are only few query points, we might not reach the desired performance and we have to refine the generated movement by means of RL.

3. Policy learning in constrained domain

The general goal of policy gradient learning is to optimize the policy parameters $\boldsymbol{\theta} \in \mathbb{R}^k$, maximizing the expected return of the state value cost function

$$J(\boldsymbol{\theta}) = E\left[\sum_{k=0}^{H} a_k r_k(\boldsymbol{\theta})\right], \tag{7}$$

where k is the time step, a_k are time step- dependent weighting factors, H is the horizon which can be infinite and r_k is the reward received at each time step. It has become a widely accepted alternative to the value function-based RL procedure.[13] Here, we assume that our task can be described as an episodic task. The general parameter update rule of the policy gradient methods, which follows the steepest descent on the expected return, is

$$\boldsymbol{\theta}_{m+1} = \boldsymbol{\theta}_m + \alpha_m \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}), \tag{8}$$

where α_m denotes the learning rate. If the gradient estimate is unbiased and the learning rate fulfills $\sum_{m=0}^{\infty} \alpha_m > 0$ and $\sum_{m=0}^{\infty} \alpha_m^2 = \text{const}$, the learning process is guaranteed to converge at least to a local minimum.[14] One of the most important advantages of the policy gradient methods over the traditional RL techniques is that we can easily limit and control the update steps. Namely, a drastic change of parameters can be hazardous for the robot and its environment. Additionally, drastic changes make the initialization of the policy based on domain knowledge or imitation learning useless, as the initial parameters can vanish after a single update step.[15]

The main problem of policy gradient methods is how to obtain a good estimator for the policy gradient $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$. If the deterministic model of the system (environment) was available, we could compute this gradient by

$$\nabla J = \frac{\partial \sum_{k=0}^{H} a_k r_k}{\partial \boldsymbol{\theta}}. \tag{9}$$

Unfortunately, such a model is normally not available; therefore, a number of policy gradient estimation methods were proposed, such as finite gradient methods,[14] likelihood ratio methods,[16] natural policy gradients,[17] etc. Policy gradient estimation becomes problematic as the dimensionality of policy parameters increases, since a large number of trials have to be performed in order to accurately estimate the gradient. However, if a sufficient amount of previous experiences is available to perform statistical generalization (as described in Section 2), we can estimate the mapping from some lower dimensional parameters $\mathbf{q}$ to the corresponding policy parameters $\boldsymbol{\theta}$. Let us assume that $\mathbf{G}(\mathbf{q}; \mathcal{Z})$ is an exact (ideal) function and $\hat{\mathbf{G}}(\mathbf{q}; \mathcal{Z})$ its approximation and that the relationship $\boldsymbol{\theta} = \mathbf{G}(\mathbf{q}; \mathcal{Z}) = \hat{\mathbf{G}}(\mathbf{q} + \Delta\mathbf{q}; \mathcal{Z})$ exists. Then, the learning problem defined in Equation (7) can be converted into

$$\hat{J}(\mathbf{q}) = E\left[\sum_{k=0}^{H} a_k \hat{r}_k(\mathbf{q})\right], \tag{10}$$

$$\hat{r}_k(\mathbf{q}) = r_k(\hat{\mathbf{G}}(\mathbf{q}; \mathcal{Z})), \tag{11}$$

and the update step (8) becomes

$$\mathbf{q}_{m+1} = \mathbf{q}_m + \alpha_m \nabla_{\mathbf{q}} \hat{J}(\mathbf{q}), \tag{12}$$

where the dimensionality of $\mathbf{q}$ is much lower than that of $\boldsymbol{\theta}$.

4. Error learning

Let us assume now that our task is of finite horizon and that we can obtain only the final reward, i.e. $r = r_H$, and that the success of our policy can be described with a vector $\boldsymbol{\varepsilon}$, which denotes the difference between the desired query $\mathbf{q}_0$ and the actually obtained query $\mathbf{q}_a = \mathbf{q}_0 + \Delta\mathbf{q} = \hat{\mathbf{G}}^{-1}(\mathbf{G}(\mathbf{q}; \mathcal{Z}); \mathcal{Z})|_{\mathbf{q}=\mathbf{q}_0}$,

$$\boldsymbol{\varepsilon} = \mathbf{q}_0 - \mathbf{q}_a. \tag{13}$$

We define the final reward as (1) $r = \boldsymbol{\varepsilon}^T\boldsymbol{\varepsilon}$ and (2) $r = \mathrm{e}^{-\boldsymbol{\varepsilon}^T\boldsymbol{\varepsilon}}$ (see Figure 1(a)). Let us compute $\nabla_{\mathbf{q}}\hat{J}$ for both cases

$$\nabla_{\mathbf{q}}\hat{J} = -2\frac{\partial \mathbf{q}_a}{\partial \mathbf{q}}\boldsymbol{\varepsilon} \approx -\mathbf{K}\boldsymbol{\varepsilon}, \tag{14}$$

$$\nabla_{\mathbf{q}}\hat{J} = 2\mathrm{e}^{-\boldsymbol{\varepsilon}^T\boldsymbol{\varepsilon}}\frac{\partial \mathbf{q}_a}{\partial \mathbf{q}}\boldsymbol{\varepsilon} \approx \mathrm{e}^{-\boldsymbol{\varepsilon}^T\boldsymbol{\varepsilon}}\mathbf{K}\boldsymbol{\varepsilon}, \tag{15}$$

where the matrix $\mathbf{K}$ is an approximation for the partial derivative $\frac{\partial \mathbf{q}_a}{\partial \mathbf{q}} = \frac{\partial}{\partial \mathbf{q}}\left(\hat{\mathbf{G}}^{-1} \circ \mathbf{G}\right)\Big|_{\mathbf{q}=\mathbf{q}_0}$ around the desired query point. Note that in the case of the ideal generalization, i. e. $\mathbf{G} = \hat{\mathbf{G}}$, this matrix is an identity matrix. In practice, the matrix $\mathbf{K}$ can be approximated using numerical derivatives of the generalization function. Taking into account that in the case of cost function (14) we have to minimize the cost while in the case of (15) we have to maximize it, we obtain the following two update rules:

$$\mathbf{q}_{m+1} = \mathbf{q}_m + \mathbf{K}\boldsymbol{\varepsilon}, \tag{16}$$

and

$$\mathbf{q}_{m+1} = \mathbf{q}_m + \mathrm{e}^{-\boldsymbol{\varepsilon}^T\boldsymbol{\varepsilon}}\mathbf{K}\boldsymbol{\varepsilon}. \tag{17}$$

The first update rule is the well-known error-based learning algorithm [8] and the second is the weighted version of the error-based learning. The second algorithm weights ε and prevents large parameter updates at large $\boldsymbol{\varepsilon}$, as shown in Figure 1(b).

Another representation of the error-based learning in one dimension of $\mathbf{q}$ is shown in Figure 2. Assuming the ideal mapping $\hat{\mathbf{G}}(\mathbf{q}; \mathcal{Z}) = \mathbf{G}(\mathbf{q}; \mathcal{Z})$ as defined by Equation (6), every desired query $\mathbf{q}$ maps to the actual query $\mathbf{q}_a$. In real situations, the generalization is seldom ideal. Figure 2 shows three cases, (a), (b) and (c), where an ideal mapping is represented with a black line and actual (real) mapping is a green curve. In the case (a), actual mapping $\mathbf{q} \longmapsto \mathbf{q}_a$ is a continuous function and the gradient has the same sign as the ideal mapping. This is the most favorable case for the error-based learning. The learning algorithm searches in the direction defined by the ideal mapping. The blue circle denotes actual query $\mathbf{q}_a$, red circle the desired query $\mathbf{q}$, and the yellow circle the solution found using the search along the ideal mapping. In the case (b), the actual mapping is again a continuous function, but the gradient of the actual mapping changes the sign. This implies that the mapping

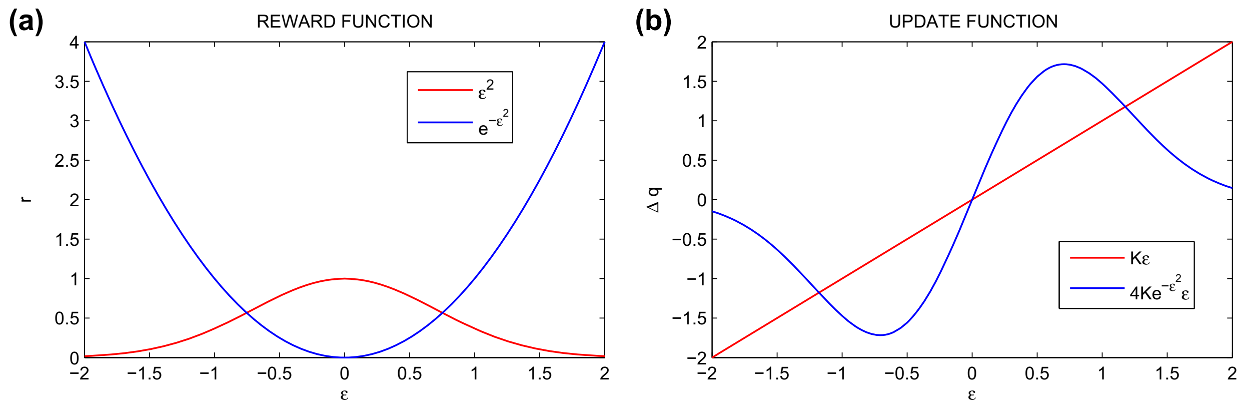

Figure 1. Reward and update function for both cases.

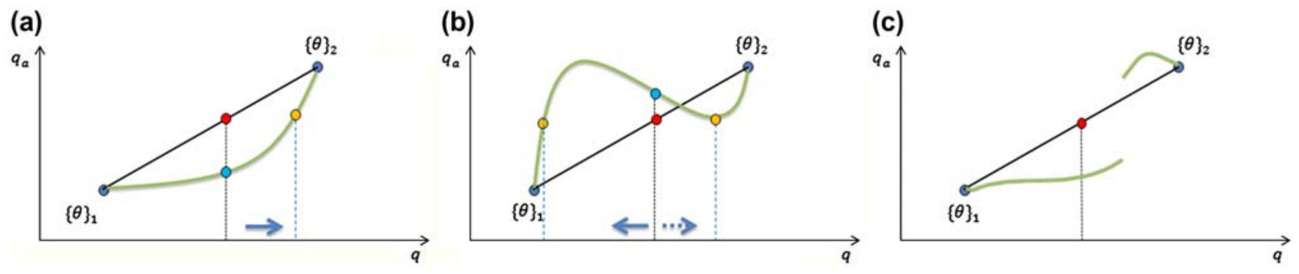

Figure 2. Mapping of the desired query to the actual query.

$\mathbf{q} \longmapsto \mathbf{q}_a$ is not unique; we might have multiple solutions. Nevertheless, error-based learning can still find the solution, but this might not be the closest one. The third case (c) shows the situation where the real mapping is not a continuous function and where the search along the ideal mapping does not lead to the desired solution-obtaining the desired $\mathbf{q}_a$. Note that the generalization and thus the error-based learning always returns a solution, but this solution does not necessary map to the desired actual query. This is the case where the learning in constrained domain leads only to an approximate solution. In such a case, we should proceed with the policy learning in the full parameter space $\boldsymbol{\theta}$, as it was proposed in [7].

Error learning requires appropriately chosen gain matrix $\mathbf{K}$, which might vary from case to case. For an autonomous, robot this is a problem because we cannot afford to guess the learning parameters for each specific case. Moreover, fixed gain is not appropriate since the mapping $\mathbf{q} \longmapsto \mathbf{q}_a$ is not linear as a rule (see Figure 2). If this mapping was linear, then the statistical model used for generalization would already be perfect and we would not need any learning at all.

Error learning can be interpreted as a problem of finding the zero of a unimodal function, which enables us to use the well-known line-search algorithms.[9] The line-search approach starts with a decent direction along which the objective function $\mathbf{q}_a$ should be minimized and then computes a step size that decides how far $\mathbf{q}$ should move along that direction. The descent direction can be computed by various methods, such as gradient descent, Newton's method and Quasi-Newton method.[9] The majority of line search approaches require the computation of the value of the objective function $\mathbf{q}_a$ at each iteration step, which is not appropriate for sensorimotor learning. Since we do not have any initial model of the $\mathbf{q} \longmapsto \mathbf{q}_a$, we would have to perform one trial for each iteration, which would be very time consuming. In order to minimize the number of trials, we use the golden section search algorithm for finding the zero of the function $\boldsymbol{\varepsilon}$.[18] It requires only a single trial for each optimization step.

Algorithm for error learning with golden section search

```
set search limits $q_l$ and $q_h$
set desired query $q_0$
set $q = q_0$
repeat
    generalize trajectory for query $q$
    execute trajectory, get $q_a$
    calculate error $\boldsymbol{\varepsilon} = q_0 - q_a$
    if = $\boldsymbol{\varepsilon} > 0$
        $q_l = q$
        $q = q_l + (q_h - q_l) * 0.618$
    else
        $q_h = q$
        $q = q_l + (q_h - q_l) * 0.382$
until $\boldsymbol{\varepsilon} <$ desired precision
```

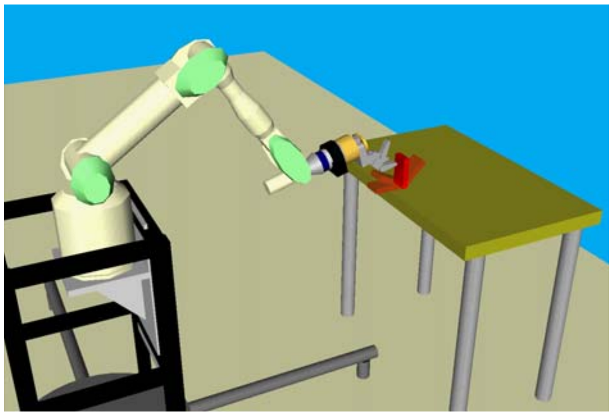
Figure 3. Simulated environment for the matchbox flip-up learning.

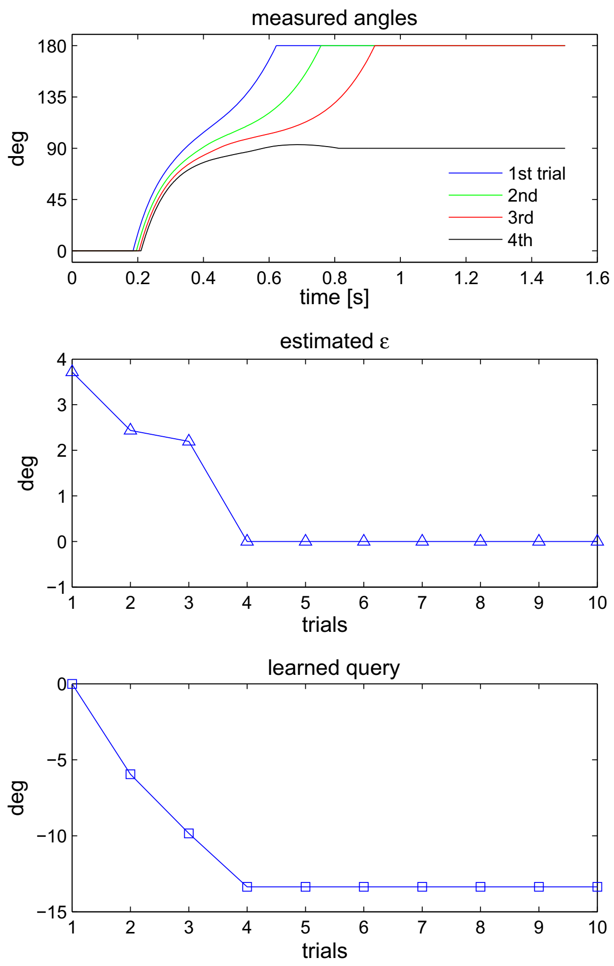

Figure 4. Angles and error convergence of the matchbox flip-up learning in simulation.

5. Simulation and experimental results

The proposed learning approach was evaluated both in simulation and on a real robot. Two case studies were carried out: the first case was to flip up matchbox to stand upright on a narrower side. The second case was learning of a barman skill, where the robot had to learn how to pour an equal quantity of the liquid into a glass from bottles containing different volumes.

For the first experiment, we used the seven degrees of freedom of a Mitsubishi Pa-10 robot equipped with Barrett hand and a vision system for the matchbox tracking at 120 Hz. For simulation, we used our own Matlab Simulink simulation system.[19] Human- demonstrated trajectories were captured by the *Optotrak* motion capture system, where we measured the wrist motion. The middle finger motion was captured using *Cyber* glove. Note that matchbox flip-up is a very challenging task even for humans. Most humans can succeed to do it only occasionally. For this reason, we recorded two unsuccessful attempts. In the first attempt, the matchbox flipped back and in the second, it flipped over. Every example trajectory $\mathcal{Z}$ was generated by mapping the recorded Cartesian human wrist motion to the robot wrist motion and the human middle finger motion to the joint motion of the middle finger of Barret Hand. Since we had only two demonstration trajectories, generalization problem defined in Equation (6) was reduced to the simple linear interpolation.

Next, we defined the reward function. Intuitively, when the matchbox flips back, the reward is given upon how close it came to the upright position by measuring the angle. When it flips forward, the reward is assigned upon the angular velocity in the upright position, which is identical to the angle measured few instances after the matchbox has reached the upright position. Thus, the error ε is defined as

$$\varepsilon = \begin{cases} \varphi_{\max} - \frac{\pi}{2} & \text{if flipped back} \\ \varphi(t_0 + \Delta t) - \frac{\pi}{2} & \text{if flipped over} \end{cases} \tag{18}$$

where $\varphi_{\max}$ is the maximal angle that the matchbox has reached, t_0 is the time when the matchbox has reached the upright position, and Δt a suitably chosen time interval, in this task 0.05 s.

First, we tested the learning in simulation. The dynamics of the matchbox flip was simulated using ODE.[20] Figure 3 shows an instance of the simulated task. We applied the error-based learning of Equation (16) to optimize the task. The gain $\mathbf{K}$ was set to the identity matrix $\mathbf{I}$. In a simulated environment, the robot succeeded to learn the appropriate policy for matchbox flip-up task in four trials. Figure 4 shows error convergence. The same experiment was repeated also on the real robot, as shown in Figure 5. The results are shown in Figure 6. Due to noise introduced by the real hardware into the system, the error oscillates around 0 until it reaches a steady state 0 in five trials.

The desired query can lie inside or outside the interval defined by the queries of the example trajectories. Most generalization techniques are able to generalize in both cases, although the performance deteriorates as the query moves further away from the example interval. Consequently, our learning technique can be applied to both cases. To show this, we simulated the same task using demonstrations where

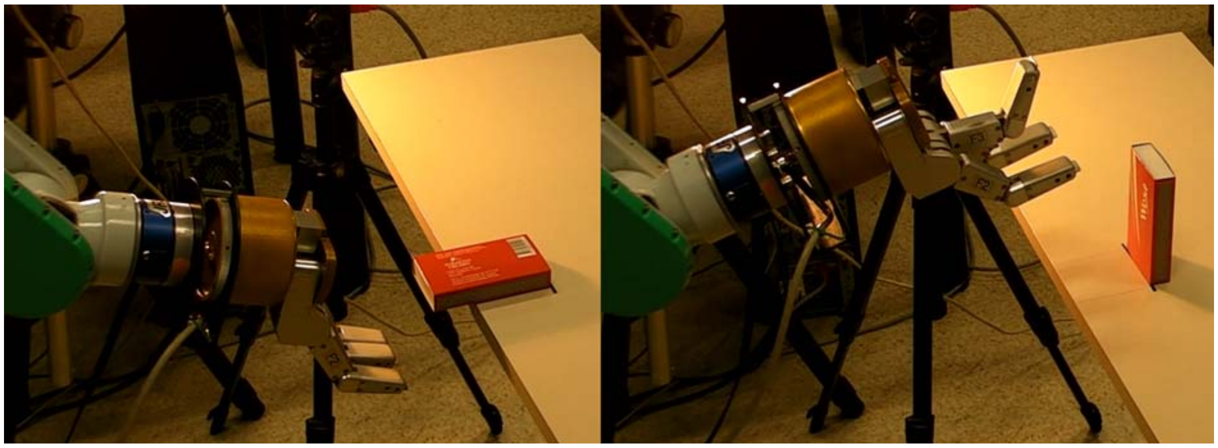
Figure 5. Learning of matchbox flip-up with the real robot.

the matchbox has flipped back in both demonstrations. Also, in this case, the robot succeeds to learn the appropriate policy, as we can see in Figure 7. In general, the proposed learning method requires only a rough estimation for the direction of the parameter update and this direction is estimated from the demonstration trajectories and the associated queries.

The second experiment was conducted on a humanoid torso composed of two KUKA LWR arms with seven degrees of freedom, Barret hands, and seven degrees of freedom humanoid head with foveal and peripheral stereo vision, as shown in Figure 8. The task was to learn how to pour equal quantity of liquid into a glass from bottles containing different volumes of liquid. We provided two demonstration trajectories, wherein both cases 0.2 l of liquid

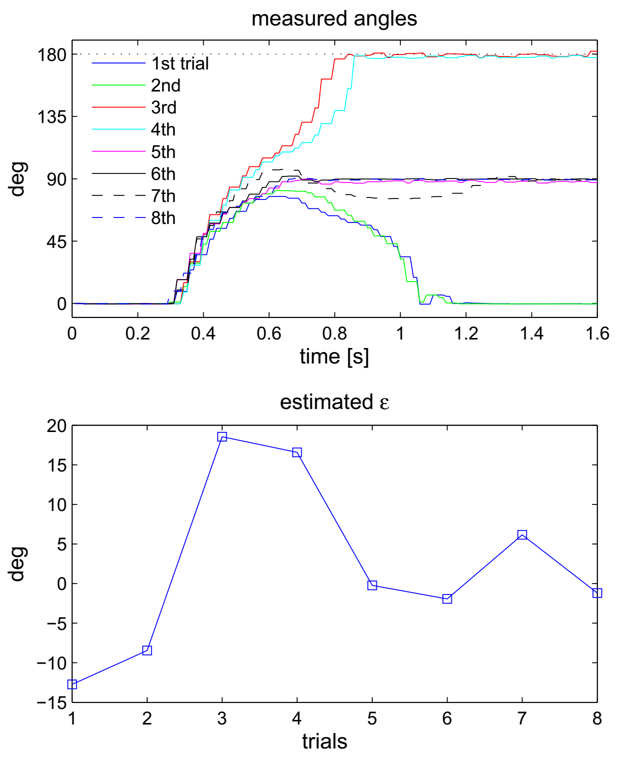

Figure 6. Estimated angles and error convergence of the matchbox flip-up learning on the real robot.

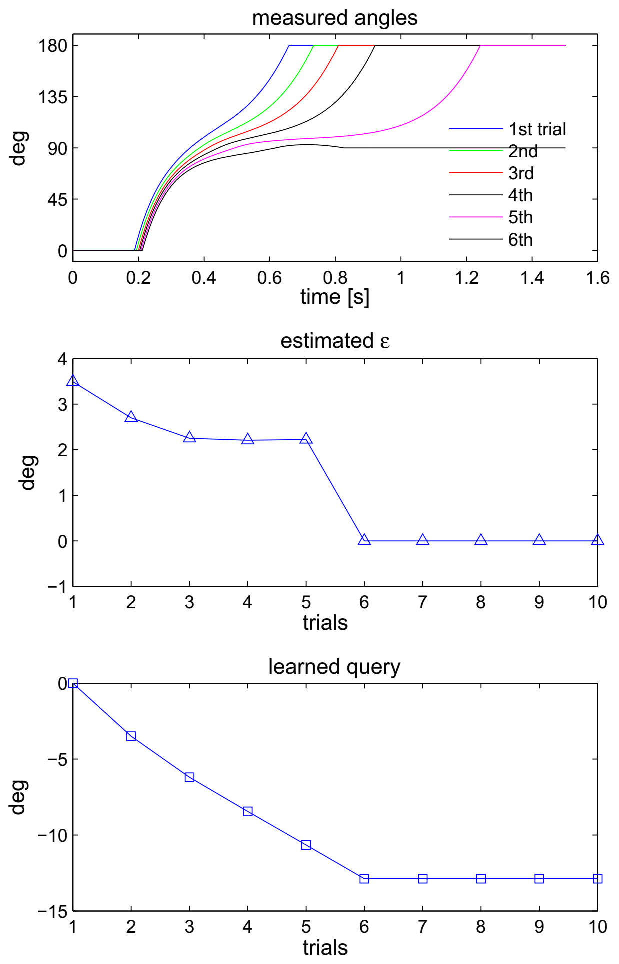

Figure 7. Angles and error convergence of flip-up learning in simulation. The demonstration trajectories caused matchbox to flip to the same side.

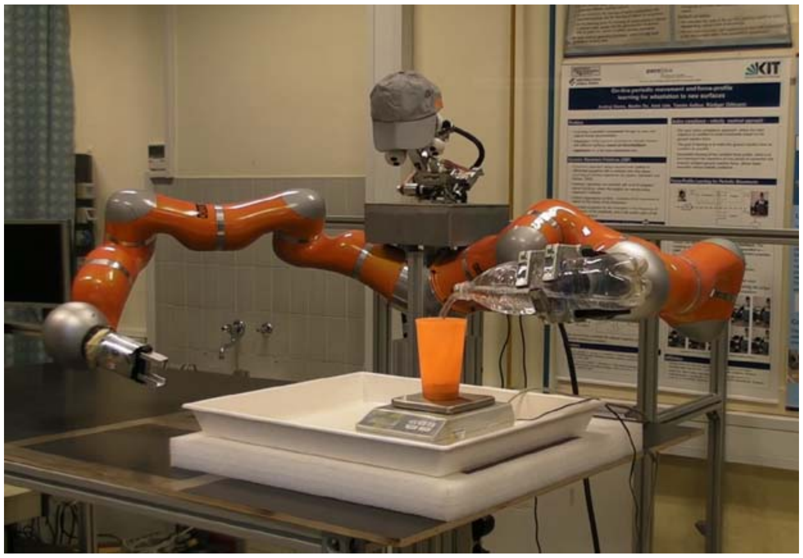

Figure 8. Real pouring setup.

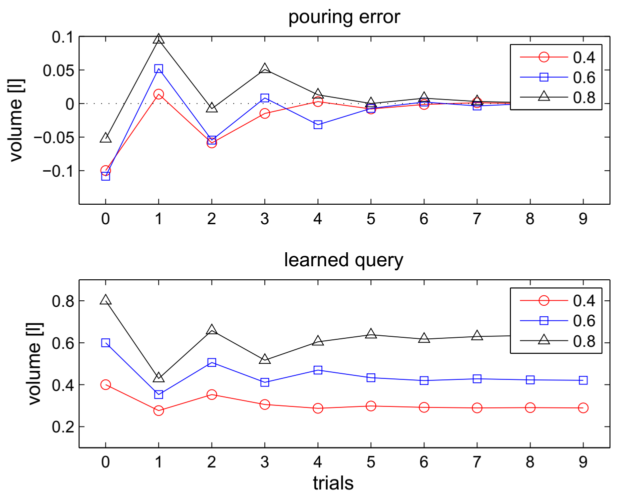

Figure 9. Convergence of the simulated pouring error and the learned query for different volumes in the bootle.

was poured into the glass from a bottle containing 0.3 l and 1.0 l of liquid, respectively. Demonstration trajectories $\mathcal{Z}$ were obtained using kinesthetic guiding, where the demonstrator manually guided the robot in a zero-gravity mode. Joint trajectories were captured directly by reading the robot sensors, mapped into Cartesian coordinates and encoded as DMPs using Equations (1)–(4). The quantity of the poured liquid was measured with a precision scale. The task was to learn how to pour 0.2 l of liquid from a bottle containing 0.5 l of liquid. For this case, error learning with the fixed gain was less successful. It required much higher gains for learning to pour from the bottles containing lower volumes of liquid compared to the bottles containing higher volumes. For this reason, we applied golden section search algorithm to speed up learning. Figure 9 shows the convergence of learning in simulation. A snapshot of the simulated environment is shown in Figure 10. The golden section cut algorithm assured similar learning rates regardless of the query point, which is the volume of liquid in the bottle this case. The same experiment was repeated using the real robot. Figure 11 shows the resulting convergence and the learned query for this case. Note the extremely fast learning rate; the robot was able to learn the appropriate policy in just few trials.

Video of our experiments is available at http://www.ijs.si/usr/aude/ar.mov

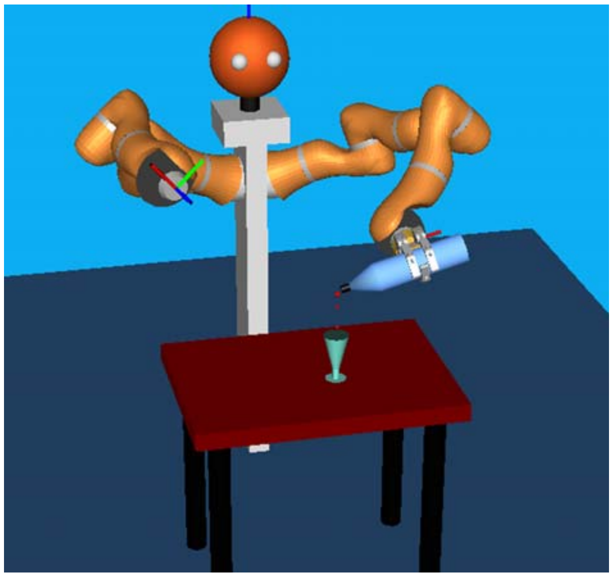

Figure 10. Simulation of pouring.

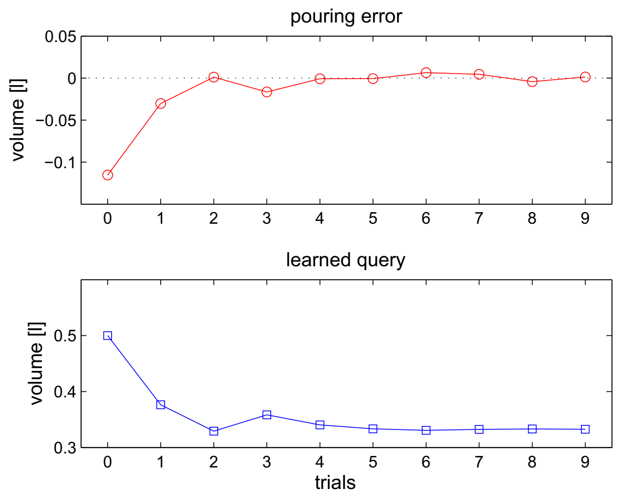

Figure 11. Convergence of the pouring error and the learned query on the real robot.

6. Conclusions

In this paper, we presented a novel approach to skill learning that combines ideas from the statistical generalization and error learning. In the first stage, we generalize the available training data to compute a control policy suitable for the current situation. This initial approximation is further improved using learning on the subspace defined by the training data, which results in learning in a state space of reduced dimensionality. The main advantage of the proposed algorithm is that the direction of the parameter update is computed from the initial database containing the demonstration trajectories. For the same reason, we can apply error learning, which turns out to be the most effective learning method when at least approximate direction of parameter update is available. The proposed approach was verified both by simulation and on the real robot for the matchbox flip-up task and for learning how to pour from bottles containing different volumes of liquid. Simulation, as well as the experimental results, exhibits exceptionally fast learning rate of the proposed approach. In our experiments, we demonstrated that the robot can learn the desired policy with very sparse statistical knowledge encoded as $\mathcal{Z}$. In both experiments only two demonstration trajectories sufficed for learning the generalization to the given tasks. In general, the richer the initial knowledge is, the fewer learning steps are required for generalization to a new case. Moreover, the robot could add newly learned trajectories together with the associated queries to the data-base, thus speeding up the learning for new situations as they arise. For example, for the second experiment the robot will require fewer and fewer learning steps to learn how to pour different levels of liquid into a glass from bottles containing different volumes of liquid.

Acknowledgement

The research leading to these results has received funding from the European Community's Seventh Framework Programme FP7/2007-2013 (Programme and Theme: ICT-2011.2.1, Cognitive Systems and Robotics) under grant agreement no. 600578, ACAT.

Notes on contributors

Bojan Nemec is senior research associate at Dept. of Automatics, Biocybernetics and Robotics, Jožef Stefan Institute. He received BS, MSc and PhD degree from the Univerity of Ljubljana. He spent his sabbatical leave at the Institute for Real-Time Computer Systems and Robotics, University of Karlsruhe. His research interests include robot control, robot learning, sensor guided control, service robots and biomechanical measurements in sport. He has published over 150 conference and journal papers and co-author of a book.

Rok Vuga received a master's degree at Faculty of Electrical Engineering, University of Ljubljana, Slovenia in October 2011. He is currently pursuing a PhD at University of Ljubljana. He holds a Young Researcher position at Humanoid and Cognitive Robotics Lab, Dept. of Automatics, Biocybernetics and Robotics, Jožef Stefan Institute, Ljubljana. He was a guest researcher at ATR Computational Neuroscience Laboratories in summer 2012 and spring 2013. His research interests include knowledge extraction from observation, reinforcement learning, and balancing for humanoids.

Aleš Ude received the Diploma degree in applied mathematics from the University of Ljubljana, Slovenia, and the PhD degree from the Faculty of Informatics, University of Karlsruhe, Germany. He was awarded the Science and Technology Agency fellowship for postdoctoral studies in ERATO Kawato Dynamic Brain Project, Japan. He is currently the head of Humanoid and Cognitive Robotics Lab, Department of Automatics, Biocybernetics, and Robotics, Jožef Stefan Institute, Ljubljana. He is also associated with the ATR Computational Neuroscience Laboratories, Kyoto, Japan. His research interests include autonomous robot learning, imitation learning, humanoid robot vision, perception of human activity, humanoid cognition, and humanoid robotics in general.

References

[1] Theodorou EA, Buchli J, Schaal S. A generalized path integral control approach to reinforcement learning. J. Mach. Learn. Res. 2010;11:3137–3181.

[2] Kober J, Peters J. Policy search for motor primitives in robotics. Mach. Learn. 2011;84:171–203.

[3] Scholz JP, Schöner G. The uncontrolled manifold concept: identifying control variables for a functional task. Exp. Brain Res. 1999;126:289–306.

[4] Kormushev P, Calinon S, Saegusa R, Metta G. Learning the skill of archery by a humanoid robot iCub. In: Proceedings – IEEE-RAS International Conference on Humanoid Robots; Nashville, TN; 2010.

[5] Grollman DH, Billard A. Donut as I do: Learning from failed demonstrations. In: Proceedings – 2011 IEEE International Conference on Robotics and Automation; Shanghai, China; 2011. p. 3804–3809.

[6] Kober J, Wilhelm A, Oztop E, Peters J. Reinforcement learning to adjust parametrized motor primitives to new situations. Auton. Robot. 2012;33:361–379.

[7] Nemec B, Vuga R, Ude A. Exploiting previous experience to constrain robot sensorimotor learning. In: 2011 11th IEEE-RAS International Conference on Humanoid Robots; Bled, Slovenia; 2011. p. 727–723.

[8] Wolpert DM, Diedrichsen J, Flanagan JR. Principles of sensorimotor learning. Nat. Rev. Neurosci. 2011;12:739–751.

[9] Nocedal J, Wright S. Numerical optimization. New York: Springer; 1999.

[10] Ijspeert AJ, Nakanishi J, Hoffmann H, Pastor P, Schaal S. Dynamical movement primitives: learning attractor models for motor behaviors. Neural Comput. 2013;25:328–373.

[11] Ude A, Gams A, Asfour T, Morimoto J. Task-specific generalization of discrete and periodic dynamic movement primitives. IEEE Trans. Robot. 2010;26:800–815.

[12] Rasmussen CE, Williams C. Gaussian processes for machine learning. Cambridge (MA): MIT Press; 2006.

[13] Sutton R, Barto A. Reinforcement learning: an introduction. Cambridge (MA): MIT Press; 1998.

[14] Peters J, Schaal S. Reinforcement learning of motor skills with policy gradients. Neural Networks. 2008;21:682–697.

[15] Schaal S. Is imitation learning the route to humanoid robots? Trends Cogn. Sci. 1999;3:233–242.

[16] Williams RJ. Simple statistical gradient-following algorithms for connectionist reinforcement learning. Mach. Learn. 1992;8:229–256.

[17] Kakade SA. Natural policy gradient. Adv. Neural Inf. Process. Syst. 2002;14:1531–1538.

[18] Teukolsky W, Vetterling S, Flannery W. Numerical recipes: the art of scientific computing. 3rd ed. New York: Cambridge University Press; 2007.

[19] Nemec B, Žlajpah L. Modeling of robot learning in Matlab/Simulink environment. In: Proceedings of EUROSIM International Conference; Prague, Czech Republic; 2010 Sep.

[20] Smith R. Open dynamic engine. 2001. Available from: http://www.ode.org