2012 12th IEEE-RAS International Conference on Humanoid Robots
Nov.29-Dec.1, 2012. Business Innovation Center Osaka, Japan

Applying statistical generalization to determine search direction for reinforcement learning of movement primitives

Bojan Nemec, Denis Forte, Rok Vuga, Minija Tamošiūnaitė+, Florentin Wörgötter+, Aleš Ude
Jožef Stefan Institute, Department of Automatics, Biocybernetics, and Robotics, Ljubljana, Slovenia,
Bernstein Center for Computational Neuroscience, Göttingen, Germany+,
bojan.nemec@ijs.si,denis.forte@ijs.si,rok.vuga@ijs.si,m.tamosiunaite@if.vdu.lt,
worgott@bccn-goettingen.de,ales.ude@ijs.si

***Abstract*—In this paper we present a new methodology for robot learning that combines ideas from statistical generalization and reinforcement learning. First we apply statistical generalization to compute an approximation for the optimal control policy as defined by training movements that solve the given task in a number of specific situations. This way we obtain a manifold of movements, which dimensionality is usually much smaller than the dimensionality of a full space of movement primitives. Next we refine the policy by means of reinforcement learning on the approximating manifold, which results in a learning problem constrained to the low dimensional manifold. We show that in some situations, learning on the low dimensional manifold can be implemented as an error learning algorithm. We apply golden section search to refine the control policy. Furthermore, we propose a reinforcement learning algorithm with an extended parameter set, which combines learning in constrained domain with learning in full space of parametric movement primitives, which makes it possible to explore actions outside of the initial approximating manifold. The proposed approach was tested for learning of pouring action both in simulation and on a real robot.**

I. Introduction

Autonomy is one of the main unresolved issues in contemporary robotics. In order to create fully autonomous robots, efficient and robust learning algorithms are indispensable. This is especially true for humanoid robots with many degrees of freedom. Among the most promising paradigms are imitation learning and reinforcement learning (RL). Robot actions are often encoded using parameteric representations with a large number of parameters, thus the search space that reinforcement learning algorithms need to explore is normally very large. Recently, new probabilistic algorithms such as PI^2 [1] and PoWER [2] were developed to deal with sensorimotor learning in high dimensional spaces. Despite of these advances, learning capabilities of modern robots are still far from the learning capabilities of humans. While humans can quickly adapt to new situations, robots often have to re-learn the whole policy in a lengthy exploration process, even when a good initial policy approximation is 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. Approaches dealing with the problem of how to make learning more efficient often rely on reducing the number of parameters to be learned.

It has been argued that in many cases the number of parameters relevant for the task is rather small [3]. For example, Scholz and Schöner [4] studied the stand-up task in humans and found that the center of mass is among the most relevant parameters. Ude et al. [5] generalized tasks such as drumming, ball throwing, and reaching with respect to a small number of characteristic parameters. Kormushev et al. [6] dealt with archery skill using a humanoid robot and suggested an algorithm, where the parameter update is formed as a linear combination of parameters reweighed according to the reward in previous roll-outs. Grollman and Billard [7] proposed a learning method, where the search space is constrained to the area between two unsuccessful demonstrations. Kober et al. [8] developed an algorithm where the control policy was improved by adapting a small set of global parameters, called meta-parameters.

In our previous work [9] we exploited previous experience to generalize to new policies and limit the search space for reinforcement learning. The goal of this paper is to show that the generalization function can provide a good initial guess for the most promising search direction. Instead of directly searching in the space of all policy parameters, the proposed method generates the appropriate search direction from previous examples. We show, that reinforcement learning in constrained domain can be represented as an error learning algorithm [10]. In this case, the learning problem turns into a problem of finding the zero of a unimodal function, which enables us to use the well developed theory of line-search algorithms [11], thus minimizing the number of roll-outs required to learn the desired sensorimotor policy. We evaluate these ideas on an upper-body humanoid robot, where the robot has to learn how to pour a given quantity of liquid into a glass from bottles containing different amounts of liquid.

The paper is organized as follows. In the next section we briefly present a statistical method for generalization of actions from previous examples, where we used dynamic movement primitives as the underlying movement representation. In Section III we introduce DMP parameter learning in constrained domain using policy gradient reinforcement learning. In the following section the error learning algorithm is evaluated. Next section introduces simultaneous parameter learning in the constrained domain and full unconstrained domain by extending the parameters set. The paper concludes with results obtained in simulation and on a real robot.

II. Action Generalization from Previous Experiences

We start with a set of robot movements $\mathcal{Z} = \{\mathbf{M}_i\}_{i=1}^{NumEx}$, where each movement $\mathbf{M}_i$ results in a successful execution of the task in a specific situation. The problem of generalization to new situations can be solved if we can characterize the task by a small number of characteristic parameters. Lets denote these parameters, which are also called query points, by $\mathbf{q}_i \in \mathbb{R}^m$, $i = 1, \ldots, NumEx$, where m is the dimensionality of $\mathbf{q}$. For this paper it is not important how these initial movements are acquired, any standard method from robotics can be used (manual programming, programming by demonstration, reinforcement learning). To become able to accomplish a task in any given situation, the robot needs to learn a function that maps the query points $\mathbf{q}$ into the parameters describing the required movement $\mathbf{M}$, i. e.

$$\mathbf{G}(\cdot;\mathcal{Z}) : \mathbf{q} \longmapsto \mathbf{M}. \quad (1)$$

In the past we proposed different statistical approaches that can be used to learn such a function [5], [12]. In all these studies we encoded the robot movements $\mathbf{M}$ by dynamic movement primitives (DMPs) [13].

III. Policy Learning in Constrained Domain

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

$$J(\theta) = E\left[\sum_{k=0}^{H} a_k r_k(\theta)\right], \quad (2)$$

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. θ are the parameters describing the selected movement representation, e. g. dynamic movement primitives. In all interesting cases, the dimensionality n of the policy parameter space θ is significantly larger than the dimensionality m of query points. Policy gradient learning is a widely accepted alternative to the value function-based reinforcement learning [14]. Here we assume that our task can be described as an episodic task. Policy gradient methods follow the steepest descent of the expected return and the general parameter update rule becomes

$$\theta_{m+1} = \theta_m + \alpha_m \nabla_\theta J(\theta), \quad (3)$$

where α_m denotes a learning rate. If the gradient estimate is unbiased and the learning rate fulfills $\sum_{m=0}^{\infty} \alpha_m > 0$ and $\sum_{m=0}^{\infty} \alpha_m^2 = const$, then the learning process is guaranteed to converge at least to a local minimum [15]. One of the most important advantages of policy gradient methods over traditional reinforcement learning techniques is that we can easily control the size of the update step. This is important because a drastic change of parameters can be hazardous both for the robot and for 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 [16].

Policy gradient methods require a good estimator for the policy gradient $\nabla_\theta J(\theta)$. If a deterministic model of the system was available, we could compute this gradient by

$$\nabla J = \frac{\partial \sum_{k=0}^{H} a_k r_k}{\partial \theta} \quad (4)$$

Unfortunately, such a model is normally not available, therefore a number of policy gradient estimation methods were proposed, such as finite gradient methods [15], likelihood ratio methods [17], natural policy gradients [18], etc. Policy gradient estimation becomes more difficult as the dimensionality of policy parameters increases, since a large number of roll-outs has to be performed in order to accurately estimate the gradient. However, if a sufficient amount of previous experiences is available to perform statistical generalization (like described in Section II), we can estimate a mapping from some lower dimensional parameters $\mathbf{q}$ to the corresponding policy parameters θ. Let assume that $\mathbf{G}(\mathbf{q};\mathcal{Z})$ is an exact (ideal) generalization function, $\hat{\mathbf{G}}(\mathbf{q};\mathcal{Z})$ its approximation, and that the relationship $\mathbf{G}(\mathbf{q};\mathcal{Z}) = \hat{\mathbf{G}}(\mathbf{q} + \Delta\mathbf{q};\mathcal{Z})$ exists for some $\Delta\mathbf{q}$. Then the learning process defined in Eq. (3) transforms into

$$\mathbf{q}_{m+1} = \mathbf{q}_m + \alpha_m \nabla_{\mathbf{q}} J(\mathbf{q}), \quad (5)$$

$$\nabla_{\mathbf{q}} J \approx \frac{\partial \sum_{k=0}^{H} a_k r_k}{\partial \mathbf{q}}, \quad (6)$$

where the dimensionality of $\mathbf{q}$ is much lower than the dimensionality of θ.

IV. Error Learning

Now we assume that the success of our policy can be described with a vector ε, which actually denotes the difference between the desired query $\mathbf{q}_0$ and the measured query $\mathbf{q}_m$,

$$\varepsilon = \mathbf{q}_0 - \mathbf{q}_m. \quad (7)$$

We define the final reward as either $r = \varepsilon^T\varepsilon$ or $r = e^{-\varepsilon^T\varepsilon}$ (see Fig. 1a). We further assume that our task has a finite horizon, that $a_k = 1$, that we can obtain only the final reward, and that the partial derivative $\frac{\partial \mathbf{q}_m}{\partial \mathbf{q}}$ around any given point is a known constant diagonal matrix $\mathbf{K}$. Note that if generalization as given by Eq. (1) is accurate, this matrix becomes an identity matrix. In many other practical cases, matrix $\mathbf{K}$ can be approximated by a diagonal and locally constant matrix. Lets compute $\nabla_{\mathbf{q}} J$ for both cases

$$\nabla_{\mathbf{q}} J = -2\frac{\partial \mathbf{q}_m}{\partial \mathbf{q}}\varepsilon \approx -\mathbf{K}\varepsilon, \quad (8)$$

$$\nabla_{\mathbf{q}} J = -2e^{-\varepsilon^T\varepsilon}\frac{\partial \mathbf{q}_m}{\partial \mathbf{q}}\varepsilon \approx -\mathbf{K}e^{-\varepsilon^T\varepsilon}\varepsilon. \quad (9)$$

This results in the update rule 1)

$$\mathbf{q}_{m+1} = \mathbf{q}_m + \mathbf{K}\varepsilon \quad (10)$$

and 2)

$$\mathbf{q}_{m+1} = \mathbf{q}_m + e^{-\varepsilon^T\varepsilon}\mathbf{K}\varepsilon \quad (11)$$

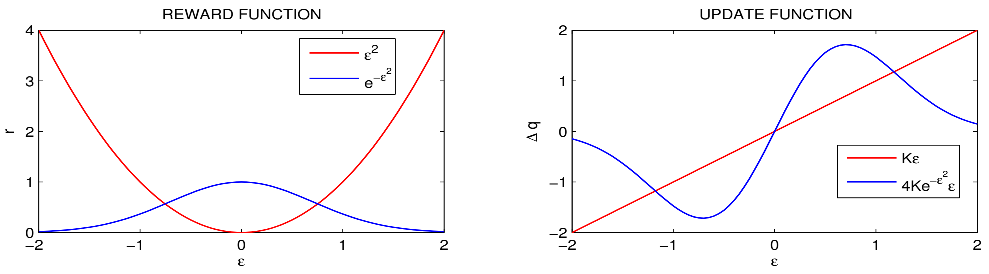

Fig. 1. Reward and update function for both cases

The first update rule is the well known error-based learning algorithm [10] and the second is its weighted version. The second update rule prevents large parameter update at large ε, as shown in Fig. 1b. Error learning requires appropriately chosen gain matrix $\mathbf{K}$, which might vary from case to case. This is a problem for autonomous learning because we can not afford to guess the learning parameters for each specific case. If this mapping was linear, the statistical model used for generalization would already be perfect and no learning would be necessary. Error learning can be represented also as a problem of finding the zero of a unimodal function, which enables us to use the well developed theory of line-search algorithms [11]. The line search finds the optimal step along the computed gradient along which the objective function $\mathbf{q}_m$ will be reduced. The majority of line search approaches require additional tuning parameters. In order to minimize the number of roll-outs and to eliminate all tuning parameters, we propose to use golden section search algorithm for finding the zero of the function ε [19], since it requires a single roll-out in each optimization step. The algorithms is explained with the following pseudo-code.

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 $q_m$
    calculate error $\varepsilon = q_0 - q_m$
    if $\varepsilon > 0$
        $q_l = q$
        $q = q_l + (q_h - q_l) * 0.618$
    else
        $q_h = q$
        $q = q_l + (q_h - q_l) * 0.382$
until $\varepsilon <$ desired precision
```

V. PoWER LEARNING IN CONSTRAINED DOMAIN

In general mapping $\mathbf{G}$ defines a low dimensional manifold of control policies and it is possible that the optimal policy for a newly observed situation is not contained in this manifold. In such a case, learning in constrained domain can only find an approximate solution. In [9] we proposed to follow the learning in constrained domain by learning in a high-dimensional space defined by all policy parameters. For example, a typical control policy encoded by DMPs requires 10 to 50 parameters for each joint, which makes the search space huge. When we switch to learning in full parameter space, the new learning process is started with a better initial guess, but no other information is reused. Here we propose how to further accelerate the learning process by combining learning in constrained domain with learning in full parameter space. For that, we assume the following process model

$$\theta = \hat{\mathbf{G}}(\hat{\mathbf{q}}; \mathcal{Z}) + \Delta\theta \tag{12}$$

Our goal is to learn such $\hat{\mathbf{q}}$ and $\Delta\theta$, which will maximize the reward r. For that, we define an extended parameter set in the form

$$\theta^* = [\hat{\mathbf{q}}, \Delta\theta]^T \tag{13}$$

Recently, efficient methods which combine the well-developed methods from statistical learning and empirical inference with classical RL approaches were proposed [1], [2]. Such algorithms can scale to significantly more complex learning systems. For our experiments we selected PoWER [2], which is a policy improvement method derived from an expectation-maximization algorithm using probability matching [20]. Unlike the policy gradient methods, it does not require the tuning of the learning rate α_m. The parameter update in PoWER follows the rule

$$\theta^*_{m+1} = \theta^*_m + \frac{\langle(\theta^*_i - \theta^*_m)r(\tau_i)\rangle_{w(\tau_i)}}{\langle r(\tau_i)\rangle_{w(\tau_i)}} \tag{14}$$

where i denotes the i-th roll-out, $r(\tau_i)$ is the positive reward accumulated from the trajectory τ in i-th roll-out and $\langle\rangle_{w(\tau_i)}$ denotes the importance sampling. Extended parameters θ^*_i used in roll-outs are selected according to the stochastic exploration policy

$$\theta^*_i = \theta^*_m + \epsilon_i$$

$$\epsilon_i \sim [\mathcal{N}(0,\sigma_1^2), \mathcal{N}(0,\sigma_2^2)]. \tag{15}$$

Noise variance σ^2 is the only tuning parameter of the PoWER algorithm, which has to be carefully selected to obtain good learning results. The value is highly dependent on the process parameters θ^*. Generally, higher values of σ^2 speed up the learning and lower values of σ^2 lead to more accurate results. As shown in the previous section, learning in constrained domain characterized by $\mathbf{q}$ benefits from faster search because the search space is considerably smaller with the proper choice of the constrained configuration space.

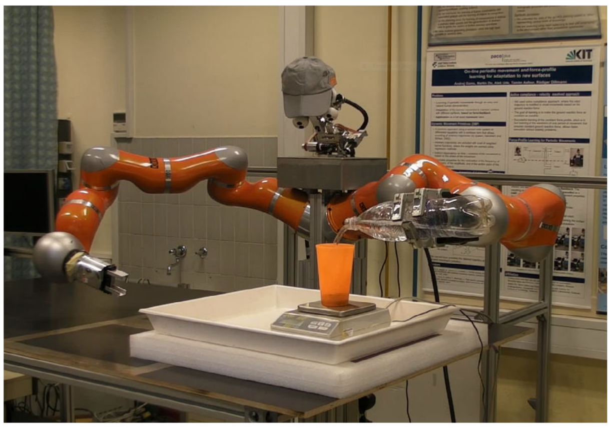

Fig. 2. Experimental setup

Thus, with proper choice of σ^2 the search in constrained space dominates over the search in unconstrained parameter space. Physical units of $\mathbf{q}$ are also considerably different from the policy parameters $\Delta\theta$, therefore we propose to use one noise variance for the query estimation and a different one for the general movement parameters.

Noise variation is difficult to guess if we start learning from scratch. However, in our system we have a number of training trajectories available, hence the appropriate σ^2 can be calculated as $\sigma^2 = s \max_{i,j}\{||\theta_i - \theta_j||\}$, where s is a suitably chosen constant, which defines the span of parameter exploration. It is good idea to decay the noise variation σ_1^2 with time, since the learning of $\hat{\mathbf{q}}$ is generally much faster than learning of $\Delta\theta$ due to the considerably different dimensions.

The update rule is the sum of all parameters sets obtained by exploration, weighted by their rewards. The role of the importance sampling is to minimize the number of roll-outs, which are needed to estimate new policy parameters. Additionally, it automatically rejects unsuccessful roll-outs, which can be caused for example by false sensor readings. It allows the RL algorithm to re-use previous most successful roll-outs τ_i during the estimation of the new policy parameters θ_{m+1}. Importance sampler sorts all past parameter updates by descending order of their rewards and rejects the less successful ones and re-weights past explorations according to θ_m [6]. In general, shorter importance sampler results in a faster learning rate and longer importance sampler in better rejection of bad parameter sets.

VI. Simulation and Experimental Results

The proposed learning approach was evaluated both in simulation and on the real robot. We tested it for learning of a barman skill, where the task is to pour the same quantity of liquid into a glass from bottles containing different volumes of liquid. The experiment was conducted on an upper-body humanoid composed of two KUKA LWR arms with 7 degrees of freedom, Barrett hands, and the Karlsruhe humanoid head [21], as shown in Fig. 2. All experiments were done also in simulation.

In our first experiment we provided two demonstration pouring movements, which poured 0.2 l of liquid into the glass from a bottle containing 0.3 l and 1.0 l of liquid, respectively. Demonstration trajectories were obtained using kinesthetic guiding and were captured both in Cartesian and joint coordinates. The quantity of liquid in the glass was measured with a precision scale after the execution of the pouring movement. No vision sensors were used to detect either the glass position or the level of liquid in the glass. With only two demonstration trajectories, the generalization algorithm from Section II is reduced to a simple linear interpolation. This gives an approximate solution, which we refined using the error learning algorithm of Section IV. Fig. 3 shows the convergence for learning of pouring movements from a bottle containing different volumes of liquid. As can be seen, the golden section cut algorithm assures similar learning rate regardless of the query point, which in this case is the volume of liquid in the bottle. The same experiment was repeated using the real robot, where the task was to pour from a bottle containing 0.5 l of liquid. Fig. 4 shows the error convergence and the learned query for this case. The robot was able to learn the appropriate policy in just a few roll-outs. Another benefit of the proposed learning algorithm is that it does not require any tuning parameters. Note also that the algorithm does not require that the demonstration trajectories are very accurate regarding the initial bottle volume or the quantity of the poured liquid. In reality, it is only required that both demonstration trajectories pour approximately equal quantity of liquid from bottles containing lower and higher volumes.

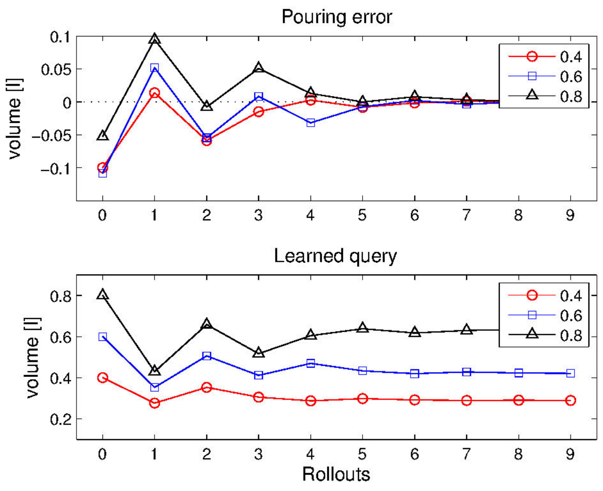

Fig. 3. Simulation results: Convergence of the pouring error and the learned query for different volumes of liquid in the bootle

In the second experiment we considered the same problem, but this time in joint coordinates and using the glass with smaller opening. Learning in joint coordinates induces additional nonlinearities in the system. In this example, the generalization in joint coordinates with only two example trajectories caused that the robot was spilling the liquid

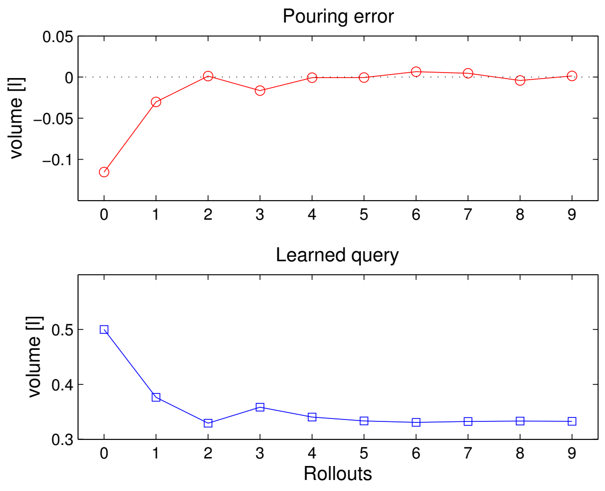

Fig. 4. Experiment with a real robot: Convergence of the pouring error and the learned query

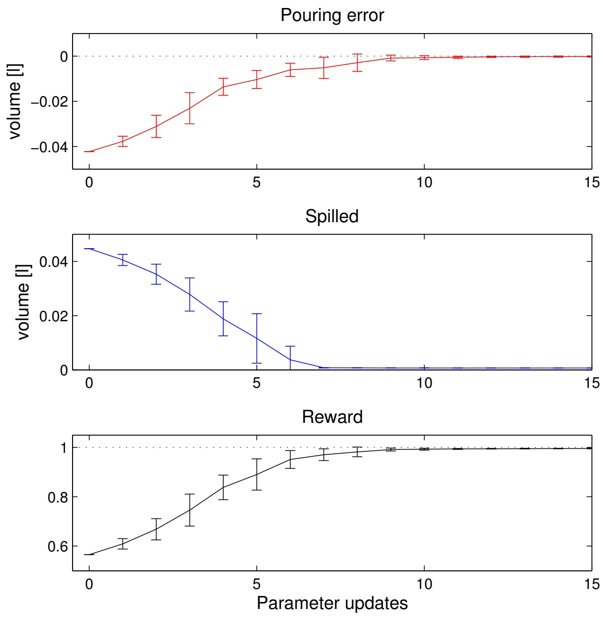

Fig. 5. Convergence of simulated pouring error, spilled volume and the reward during the joint based learning. Vertical bars shows standard deviation of 20 simulated experiments

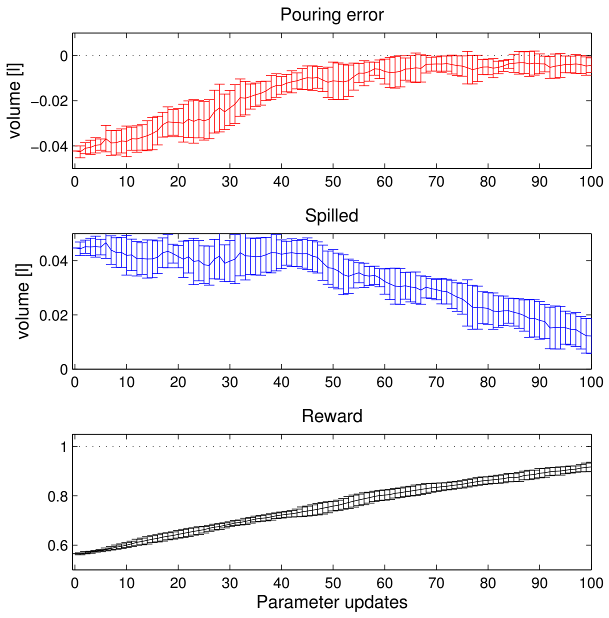

Fig. 6. Convergence of simulated pouring error, spilled volume and the reward during the joint based learning using ordinary algorithm, which searches only in unconstrained parameter space. Vertical bars shows standard deviation of 20 simulated experiments

during the execution of the pouring movement. Thus in this case the learning in constrained domain only is not appropriate and we therefore applied the full-space learning method described in Section V. The demonstrated trajectories in joint space were encoded as DMPs with 30 Gaussian kernel functions for each joint trajectory. Thus, the total number of θ^* parameters (including query, kernel weights, goals and duration) was 219. The reward function for this case was $r = 1 - 5\|v_d - v_m\| - 5v_s$, where r is the reward, v_d is the desired volume of the liquid in the glass, v_m is the measured poured volume and v_s is the volume of the spilled liquid. The constants in reward function were selected in such a way that the reward was always positive. We used three roll-outs for each parameter update and set the importance sampler length to 3. Constants s_1 and s_2 for the parameter explorations were set to 0.2 and 0.05 respectively. Fig. 5 shows the mean value of the pouring error, spilled volume, and the reward from 20 simulated learning experiments, where we learned to pour from a bottle containing 0.5 l of liquid. Note that each parameter update requires three additional roll-outs, therefore we have four roll-outs per parameter update We compared the results of the proposed algorithm with ordinary algorithm, which searches only in unconstrained parameters space. For that, we set s_1 to 0. Although both algorithms started with the same parameter set, obtained from the initial generalization, our algorithm outperforms the ordinary algorithm for a degree of magnitude, as it can be seen form Fig 6. Similar results were obtained with the real robot, where the volume of the spilled liquid was determined as the difference between liquid poured into the glass as measured with a precision scale and the drop of the bottle volume, which was estimated from the joint torque sensors of our robot. In this case we encoded the trajectory of each joint with 10 Gaussian kernel functions. The total number of learning parameters was 79. The results of the real robot experiment using the same learning parameters as simulated robot are displayed in Fig. 7. Results on a real robot are less accurate due to the noise arising from the determination of the spilled liquid quantity and the imprecise initial volume of the liquid (since the bottle needed to be refilled manually). Despite of that, learning on the real robot turned out to be faster than learning in the simulated environment. This is due to the smaller number of learning parameters used in the real experiment.

In simulation, we also used a glass with an opening of half the diameter.

The learning of pouring was considered also in [22], where we learned to pour the whole bottle volume into a glass. Although the pouring problem in this paper was more demanding, we obtained a faster convergence rate due to the proposed algorithms, which combine learning in constrained and unconstrained domain.

VII. Conclusions

In this paper we presented a novel approach to reinforcement learning in robotics. It combines ideas from statistical generalization and standard reinforcement 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 manifold defined by the training data. This enables us to perform reinforcement learning in a state space of reduced dimensionality. Another advantage of the proposed algorithm is that the direction of the parameter update is inferred 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. Although efficient, error learning can not be applied to all learning problems, therefore we proposed a second approach, which combines learning in constrained domain and learning in full parameter space. The proposed approach was verified both in simulation and on the real robot for the task where we had to learn how to pour from a bottles containing different volumes of liquid. Experimental results in simulation and on the real robot demonstrate fast learning rates of the proposed

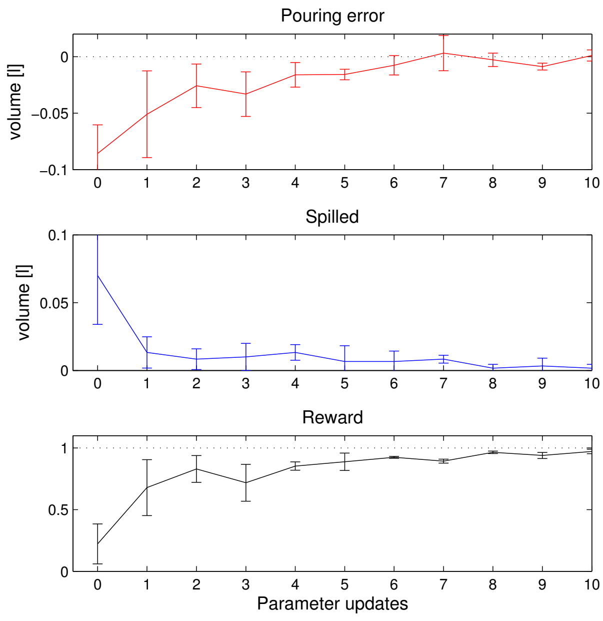

Fig. 7. Convergence of the pouring error, spilled volume and the reward during the joint based learning on the real robot.Vertical bars shows standard deviation of 5 experiments

Acknowledgment

The research leading to these results has received funding from the European Community's Seventh Framework Programme FP7/2007-2013 (Specific Programme Cooperation, Theme 3, Information and Communication Technologies) under grant agreements no. 269959, IntellAct, and no. 270273, Xperience.

References

[1] E. A.Theodorou, J. Buchli, and S. Schaal, "A generalized path integral control approach to reinforcement learning," *Journal of Machine Learning Research*, no. 11, pp. 3137–3181, 2010.

[2] J. Kober and J. Peters, "Learning motor primitives for robotics," in *Proc. IEEE Int. Conf. Robotics and Automation*, Kobe, Japan, 2009, pp. 2112 – 2118.

[3] C. G. Atkeson, A. W. Moore, and S. Schaal, "Locally weighted learning," *AI Review*, vol. 11, pp. 11–73, 1997.

[4] J. P. Scholz and G. Schner, "The uncontrolled manifold concept: identifying control variables for a functional task." *Exp. Brain Res.*, vol. 126, pp. 289–306, 1999.

[5] A. Ude, A. Gams, T. Asfour, and J. Morimoto, "Task-specific generalization of discrete and periodic dynamic movement primitives," *IEEE Trans. Robotics*, vol. 26, no. 5, pp. 800–815, 2010.

[6] P. Kormushev, S. Calinon, R. Saegusa, and G. Metta, "Learning the skill of archery by a humanoid robot iCub," in *Proc. IEEE-RAS International Conference on Humanoid Robots*, Nashville, USA, 2010.

[7] D. H. Grollman and A. Billard, "Donut as I do: Learning from failed demonstrations," in *proc. 2011 IEEE International Conference on Robotics and Automation*, Shanghai, China, 2011, pp. 3804 – 3809.

[8] J. Kober, A. Wilhelm, E. Oztop, and J. Peters, "Reinforcement learning to adjust parametrized motor primitives to new situations," *Autonomous Robot*, no. 4, pp. 361–379, 2012.

[9] B. Nemec, R. Vuga, and A. Ude, "Exploiting previous experience to constrain robot sensorimotor learning," in *2011 11th IEEE-RAS International Conference on Humanoid Robots*, Bled, Slovenia, 2011, pp. 727–723.

[10] D. M. Wolpert, J. Diedrichsen, and J. R. Flanagan, "Principles of sensorimotor learning," *Nature Reviews Neuroscience*, vol. 12, pp. 739–751, 2011.

[11] J. Nocedal and S. Wright, *Numerical Optimization*. New York: Springer, 1999.

[12] D. Forte, A. Gams, J. Morimoto, and A. Ude, "On-line motion synthesis and adaptation using a trajectory database," *Robot. auton. syst*, vol. 60, pp. 1327–1339, 2012.

[13] A. J. Ijspeert, J. Nakanishi, and S. Schaal, "Movement imitation with nonlinear dynamical systems in humanoid robots," in *Proc. IEEE Int. Conf. Robotics and Automation*, Washington, DC, 2002, pp. 1398–1403.

[14] R. Sutton and A. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA: MIT Press, 1998.

[15] J. Peters and S. Schaal, "Reinforcement learning of motor skills with policy gradients," *Neural Networks*, vol. 21, pp. 682–697, 2008.

[16] S. Schaal, "Is imitation learning the route to humanoid robots?" *Trends in Cognitive Sciences*, vol. 3, no. 6, pp. 233–242, 1999.

[17] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Machine Learning*, vol. 8, no. 23, 1992.

[18] S. A. Kakade, "Natural policy gradient," *Advances in neural information processing systems*, vol. 14, pp. 1531–1538, 2002.

[19] W. Teukolsky, S. Vetterling, and W. Flannery, *Numerical Recipes: The Art of Scientific Computing (3rd ed.)*. New York: Cambridge University Press, 2007.

[20] P. Dayan and G. Hinton, "Using expectation-maximization for reinforcement learning," *Neural Computation*, vol. 9, 1997.

[21] T. Asfour, K. Welke, P. Azad, A. Ude, and R. Dillmann, "The Karlsruhe humanoid head," in *8th IEEE-RAS International Conference on Humanoid Robots*, Daejeon, Korea, 2008, pp. 447–453.

[22] M. Tamosiumaite, B. Nemec, A. Ude, and F. Wörgötter, "Learning to pour with a robot arm combining goal and shape learning for dynamic movement primitives," *Robot. auton. syst*, vol. 59, no. 11, pp. 910–922, 2011.