2013 13th IEEE-RAS International Conference on
Humanoid Robots (Humanoids).
October 15 - 17, 2013. Atlanta, GA

Velocity adaptation for self-improvement of skills learned from user demonstrations

Bojan Nemec[1], Andrej Gams[1,2] and Aleš Ude[1]

Abstract— **We address the problem of how to increase the speed of movements that occur in contact with the environment, where the initial movements were acquired by kinesthetic guiding. We take into account dynamic capabilities and constrains of both the robot and the environment. This leads to a modified, non-uniformly accelerated motion. To enable the non-uniform modulation of the movement policy, we encode the initial control policy using an extended formulation of dynamic movement primitives. The initial policy is improved using feedback error adaptation, ILC-based learning or reinforcement learning. We propose a new policy learning algorithm which takes into account intermediate rewards during the policy learning. The proposed approach was experimentally evaluated on a bi-manual kitchen task, where the robot, composed of two KUKA LWR arms, had to assemble a cake decoration tool.**

I. Introduction

The paper deals with the problem how to autonomously improve skills obtained by learning from demonstration. Note that humans do this all the time. A teacher demonstrates a skill, often slowly, and then the subject tries to repeat it. The first few attempts might be very slow or might even fail. After that we generally try to improve the learned skill and accelerate it, i. e. speed it up. During this process of improvement, which is still essentially learning, we adapt the motion pattern to our own dynamic capabilities [1]. Speeding up is not done uniformly. Some parts of the motion pattern can be accelerated more, some less, depending on our own kinematics and dynamics. During this adaptation process, we change also the motion and force patterns and stiffness/compliance patterns. In this paper we present an algorithm that adopts this principle for robot learning and self-adaptation.

Recently there were many approaches presented on how to adapt the motion patterns learned by demonstration to the robot agent [2], [3], [4], [5]. One of the key parameters when executing tasks that involve interaction with the environment is also stiffness [6]. Human behavior is characterized by a low-gain compliant control with variable and task dependent stiffness. This is one of the reasons why in most cases humans significantly outperform robotic systems when interacting with the environment [7]. Therefore, it is natural to extend the concept of variable stiffness to robotics. Although the appropriate stiffness pattern could be demonstrated, it is questionable how to transfer the stiffness of the human demonstrator to robotics systems due to intrinsically different dynamic properties of human and robot arms. Alternatively, the impedance can be autonomously learned by means of reinforcement learning [8].

In this paper we propose a new approach to the self-adaptation and self-improvement of the learned movement policy. Besides learning the appropriate trajectory, force, and stiffness patterns, we also adapt the speed of task execution. Our approach relies on dynamic movement primitives (DMP) representation of the policy [9], which provides the ability to modify the velocity along the trajectory.

This leads to a more general approach on how to improve the motor knowledge. The overall procedure can be summarized in the following steps:

1) Acquire an initial example movement that successfully solves the given task. This can be accomplished for example by direct imitation [10] or kinesthetic guiding [11].
2) Determine the stiffness pattern (as for example in [8]).
3) Adapt the velocity pattern (this paper).
4) Record the force/torque patterns along the resulting trajectory [12].
5) Transfer the resulting trajectory to a new configuration and adapt it if to perform the recorded force/torque pattern [12].

In our approach, the execution speed is encoded as a function of the phase and learned by means of reinforcement learning. For this purpose, we propose a reinforcement learning algorithm based on PoWER, capable to take into account also intermediate rewards. This is the second contribution of this paper. We also considered iterative learning control (ILC) and feedback control approaches for the same task. The proposed methodology and algorithms were tested on a bi-manual task taken from the kitchen, where the robot has to assemble a cake decoration tool.

The paper is organized as follows. In section II we briefly outline the dynamic movement primitives, which are used as a underlying representation of motion patterns describing the policy. In Section III we present an extension to the reinforcement learning algorithm PoWER, which introduces intermediate rewards in episodic tasks. Learning of the appropriate speed profile is considered in section IV, where we

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 agreement no. 269959, IntellAct, and by Sciex-NMS^{CH} project 12.018.

[1] Department of Automatics, Biocybernetics and Robotics, Humanoid and Cognitive Robotics Lab, Jožef Stean Institute, Jamova cesta 39, 1000 Ljubljana, Slovenia, bojan.nemec@ijs.si, andrej.gams@ijs.si, ales.ude@ijs.si

[2] Biorobotics Laboratory, École Polytechnique Fédérale de Lausanne, Station 14, CH-1015 Lausanne, Switzerland

propose three approaches: 1) standard feedback control 2) an approach using iterative learning control, and 3) adaptation with extended PoWER. Experimental results in chapter V demonstrate the effectiveness of the proposed approaches for a bimanual task of assembling a cake decoration tool, which involves hard contacts. Finally, we conclude with a brief discussion on the preferred movement-policy speed adaptation method.

II. Learning by Demonstration using DMPs

Our approach relies on a parameterized policy using dynamic movement primitives [9]. Within this framework, every degree of freedom is described by its own dynamic system, but with a common phase to synchronize them. For point-to-point movements (also referred to as discrete movements), given either in joint or in task space, the trajectory of each robot degree of freedom y is described by the following system of nonlinear differential equations

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

$$\tau \dot{y} = z, \quad (2)$$

$$\tau \dot{x} = -\alpha_x x, \quad (3)$$

where x is the phase variable and z is an auxiliary variable. α_x, α_z, β_z and τ should be be defined 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 are used to modify the dynamics of the second-order differential equation system to approximate any smooth point-to-point trajectory from the initial position y_0 to the final configuration g

$$f(x) = \frac{\sum_{i=1}^{N} w_i \Psi_i(x)}{\sum_{i=1}^{N} \Psi_i(x)} x, \;\; \Psi_i(x) = \exp\left(-h_i \left(x - c_i\right)^2\right), \quad (4)$$

with the given initial velocity and final velocity equal to zero. Here c_i are the centers of radial basis functions distributed along the trajectory and $h_i > 0$ their widths. Weights w_i are estimated in such a way that the DMP encodes the desired trajectory. In the equations above, α_x, α_z, and β_z are constant. They are set so that the underlying second order linear dynamic system is critically damped [13]. Throughout this paper we use DMPs to encode Cartesian space trajectories.

One of the advantages of DMPs is that they can be modulated both spatially and temporally without changing the overall shape of motion. Ijspeert et al. [9] describe slow-down feedback, where the robot is automatically halted on excessive position error. In [12] this principle was used to slow down the trajectory execution on excessive forces and toques.

Here we propose to extend the original DMP equations (1) – (3) by an additional temporal scaling factor ν

$$\nu(x)\tau \dot{z} = \alpha_z(\beta_z(g-y)-z) + f(x), \quad (5)$$

$$\nu(x)\tau \dot{y} = z, \quad (6)$$

$$\nu(x)\tau \dot{x} = -\alpha_x x. \quad (7)$$

The scaling factor $\nu(x)$ is a function of phase and is represented by a linear combination of M radial basis functions

$$\nu(x) = \frac{\sum_{j=1}^{M} w_j \Psi_j(x)}{\sum_{j=1}^{M} \Psi_j(x)} x. \quad (8)$$

Initially, the weights w_j for ν are set so that $\nu(x) \approx 1, \; \forall \; x$, using regression. Note that here the centers and widths of radial basis functions Ψ_i are in general different than in (4). Optimal values for kernel weights w_j are determined as described in chapter IV.

III. Policy Learning with Extended PoWER

In order to make robots truly autonomous, they should be able to learn or adjust control policies based on the feedback arising from the dynamic interaction with the varying environment. The general goal of policy learning can be described as an optimization process, where the goal is to determine policy parameters $\theta \in \mathbb{R}^n$, such that they maximize the expected return of the state value cost function

$$J(\theta) = E\left[\sum_{k=0}^{H} a_k r_k(\theta)\right]. \quad (9)$$

E denotes the expectation value, k is the time step, a_k are time-step dependent weighting factors, H is the horizon, which in case of DMPs is equal to the number of sampling steps along the trajectory, and r_k is the reward received at each time step. θ are the parameters describing the selected movement representation, e. g. DMPs. There are numerous methods how to optimize the above cost function, for example finite gradient methods, likelihood ratio methods, natural policy gradients, to name just a few [14]. The main problem is high dimensionality of parameters to be learned. For example, a typical control policy encoded by DMPs might require 30 to 50 parameters for each joint, which makes the search space huge. Recently, new stochastic methods PI^2 and PoWER, which combine the well-developed methods from statistical learning and empirical inference with classical reinforcement learning (RL) algorithms, were proposed [15], [16]. Such algorithms can scale to significantly more complex learning systems and minimize the number of tuning parameters. Gradient and probabilistic learning algorithms were evaluated in [15]. It was formally and experimentally proven that PI^2 and PoWER learning algorithm perform essentially identically in cases where only terminal reward is available during the episodic policy learning. PI^2 performs better when intermediate rewards are available. On the other hand, with PoWER it is very easy to incorporate also other policy parameters [17], not just shape parameters $\mathbf{w}$, as is the case with PI^2. This motivated us to propose the extension of the PoWER algorithm for cases where at least some intermediate rewards can be obtained during the exploration of an episodic (finite horizon) task.

Let π_l be the trajectory executed in the $l-$th rolout and $\pi_{l,k}$ the part of the trajectory π_l from its beginning until the phase $x(t_k)$, where t_k is the time at index k. The parameter

update in Extended PoWER follows the rule

$$\theta_{m+1} = \theta_m + \gamma \frac{\langle (\theta_l - \theta_m)\phi(\pi_l) \rangle_{w(\pi_l)}}{\langle \phi(\pi_l) \rangle_{w(\pi_l)}} + (1-\gamma) \frac{\sum_{k=0}^{H} \frac{\langle (\theta_{l,k} - \theta_m) r(\pi_{l,k}) \rangle_{w(\pi_{l,k})}}{\langle r(\pi_{l,k}) \rangle_{w(\pi_{l,k})}} g(t_k)}{\sum_{k=0}^{H} g(t_k)}, \quad (10)$$

where $\phi(\pi_l)$ is the terminal reward accumulated in l-th roll-out, $r(\pi_{l,k})$ is the intermediate reward obtained from the trajectory $\pi_{l,k}$ in l-th roll-out at the time t_k, $k = 0, \ldots, H$, H is the number of sampling steps on the trajectory, $g(t_k)$ are the scaling factors related to the nonlinear part of the DMP defined in Eq. (4),

$$g(t_k) = \sum_{n=1}^{N} x(t_k) \Psi_n(x(t_k)), \quad (11)$$

and γ is the discount factor. $\langle \rangle_{w(\pi_l)}$ denotes the importance sampling. Parameters θ_l used in roll-outs are selected according to a stochastic exploration policy, $\theta_l = \theta_m + \epsilon_l, \epsilon_l \sim \mathcal{N}(0, \sigma^2)$. The update rule is the sum of parameter exploration in each roll-out weighted by its return.

The role of 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 π_l during the estimation of the new policy parameters θ_{m+1}. Importance sampler sorts all past parameters update by descending order of their return, and rejects less successful ones and re-weights past explorations according to previous estimate θ_m [18], [19]. Like the original PoWER, also the extended algorithm is a probabilistic policy improvement method which can be derived from an Expectation-Maximization framework using probability matching [20]. Intuitively, we can regard this algorithm as multiple PoWER algorithms that use intermediate rewards r as final rewards for trajectory sections starting at t_0 and ending at t_k, $k = 0, \ldots, H$, and weighted with basis functions $g(t_k)$. The proposed algorithm has one additional parameter γ, which weights terminal reward ϕ and intermediate rewards r. Tuning of this parameter is very intuitive, it simply denotes how much we trust the terminal and intermediate rewards, respectively. For example, value 0.5 means that they are given equal weights.

IV. Learning of the speed profile

The goal of modifying the speed profile is to accelerate the task execution, thus assuring a quicker accomplishment of the task. We assume that initially a Cartesian space trajectory, which successfully accomplishes the task, has been provided and encoded as a DMP. The main idea is simple: increase the speed of task execution until some essential task parameters are violated. In assembly tasks such a parameter might be the contact force. Contact force can be successfully detected using various force-torque sensors mounted on the robot wrist, joints, or the tool itself. Therefore it is worth considering learning of the speed profile as a feedback control problem.

A. Speed profile adaption using feedback error

In order to determine the speed profile, we first define an error function. A suitable error function to adapt the speed of motion is given by

$$e(x_k) = \xi_\nu(\nu_{\min} - \nu(x_k)) - \xi_f \|\mathbf{f}(x_k)\| - \xi_m \|\mathbf{m}(x_k)\|, \quad (12)$$

where ν is the temporal scaling factor, $\mathbf{f}$ is the vector of contact forces, $\mathbf{m}$ is the vector of contact torques, $\nu_{\min} > 0$ is the lower bound for the temporal scaling factor, x_k is the phase along the trajectory at step k, $x_k = x(t_k)$ and ξ_ν, ξ_f and ξ_τ are the corresponding weighting factors. Note that the scale factor $\nu < 1$ speeds up the task execution and $\nu > 1$ actually slows it down. The drawback of this error function is that it penalizes also forces that might be advantageous, e. g. forces required by the given assembly task. Therefore, weighting scalars ξ_f and ξ_m are defined as

$$\xi_{f,m} = \begin{cases} 0, & \text{if } \|\mathbf{f}\| < f_{\max},\ \|\mathbf{m}\| < m_{\max} \\ \xi_{f,m} & , \end{cases} \quad (13)$$

where $f_{\max}$ and $m_{\max}$ are maximal allowable force and torque norms. A simple PI control law yields the following update rule for the temporal scaling factor

$$\nu(x_k) = \begin{cases} \nu(x_{k-1}) + Ke(x_k), & \nu(x_k) > \nu_{\min} \\ \nu_{\min}, & \text{otherwise} \end{cases}, \quad (14)$$

where K is the proportional gain scalar. The error feedback approach has a serious drawback, which is that it can not prevent large force peaks due to the impact. Although these force peaks are affected by the quality of the robot trajectory and force control, they can not be canceled using only the force feedback signal. In order to prevent large force peaks arising from the impact, the robot should anticipate when they will occur. In order to solve this problem, we the apply iterative learning control framework ([21], [22]) in the next section.

B. Speed profile adaptation using ILC-based approach

Humans can acquire skills by repeating the same action over and over again. The same principle can be adopted also in machine motor control, when a system follows a similar trajectory repeatedly. In this case, the information about the tracking error can be used to improve performance in the next repetition of the same trajectory. This is the basic idea of the iterative learning control (ILC, [21], [22]). Standard ILC assumptions include: 1) Stable system dynamics, 2) System returns to the same initial conditions at the start of each trial, 3) Each trial has the same length. The third assumption could be problematic in our case, since we change the execution speed, therefore each trial has a different duration. However, we defined the temporal scaling factor ν as the function of phase in Eq. (7), thus we can sample ν the same number of times in every trial. Applying the ILC framework, the temporal scaling factor is learned as

$$\nu(x_k, j+1) = \kappa(\nu(x_k, j) + \eta_1 e(x_{k+1}, j) + \eta_2 e(x_{k+2}, j)), \quad (15)$$

where x denotes the phase, k is the time step, j is the learning iteration index and κ, η_1, and η_2 are the corresponding gains. Our implementation of ILC uses error signals $e(x_{k+1}, j)$ and $e(x_{k+2}, j)$ to anticipate the disturbance caused by the impact and compensate it with the time scale $\nu(x_k, j+1)$.

C. Speed profile adaptation using reinforcement learning

In the previous subsections we showed how to learn the temporal scaling factor ν as continuous function of phase x. In this chapter we learn ν as a parameterized policy by determining the weight vector $\mathbf{w}$ using reinforcement learning. For this purpose we apply the extended PoWER algorithm, presented in chapter III. First, we have to define the reward function, which has to be strictly positive and integrate to a finite number. A suitable choice of the intermediate reward function is

$$r(x_k) = \frac{1}{1 + |e(x_k)|}, \tag{16}$$

where $e(x_k)$ is defined with Eq. (12). Terminal reward is chosen as

$$\phi = \frac{\sum_{k=1}^{H} r(k)}{\Gamma}, \tag{17}$$

where Γ equals to 1 in the case of successful accomplishment of the task, or is a large positive value which penalizes the failure of task. Parameters $\mathbf{w}$ of the temporal scaling factor $\nu(x)$ from Eq. (8) are then learned using Extended PoWER update rule (Eq. 10), where $\theta = \mathbf{w}$. In order to reduce the computational burden and enhance the robustness of the estimation, we divided the phase of the original demonstrated trajectory $x = [1, \exp(-\alpha_x \tau)]$ into $M-1$ temporally equally spaced intervals using equation

$$x_k = \exp(-\alpha_x t_k),\ t_k = \frac{k}{M-1}\tau,\ k = 0, \ldots, M-1, \tag{18}$$

and the reward $r(x_k)$ was averaged over the phase interval $[x_{k-1}, x_{k+1}]$. τ denotes the duration of the original demonstrated trajectory and M is the number of the radial basis functions in Eq. (8), respectively. In such a way, t_k are placed at the centre of the radial basis functions and the horizon H equals to $M-1$.

V. Experimental evaluation on kitchen scenario

The proposed speed modulation methods were tested in a bimanual kitchen task, where the robot had to assemble a cake decoration tool. More precisely, it has to insert the piston into the barrel containing the creme, marmalade, etc. (see Fig. 1). The experimental setup was composed of two Kuka LWR robot arms, the left one equipped with a three finger Barret hand, and the right one with a two fingered TBK RH707 hand. The barrel was firmly held by the Barret hand, while the piston was grasped with the two fingered hand. Cake decoration injection tool was an off-the-shelf commercial product bought in the grocery store. The whole setup is shown in Fig 2.

The initial trajectory was obtained by kinesthetic guiding, where the operator freely moves the robot's tool centre point along the desired trajectory in gravity compensation mode. The trajectory was measured in Cartesian space using quaternions to encode the orientational part. Each position / orientation dimension was encoded as a single DMP with a common phase using 20 Gaussian kernel functions ($M = 20$). Note that we encoded each quaternion component as a single DMP, therefore the resulting quaternion had to be normalized after each integration step. The exact solution was proposed in [23], but since we confirmed experimentally that the differences between both approaches are negligible, we use Eq. (5, 6) also for quaternion integration. Such an approach simplifies the implementation.

Fig. 1. Cake decoration tool

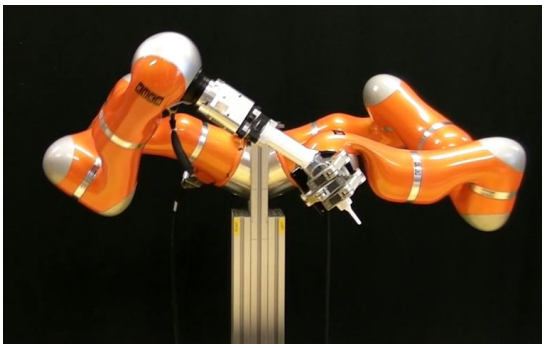

Fig. 2. Experimental setup

In this experiment we didn't learn robot stiffness parameters. The Cartesian stiffness was fixed all the time and set to $[2000, 2000, 1000]$ N/m for the positional axes and to $[300, 300, 300]$ Nm/rad for the rotational axes, respectively. The trajectory was carefully demonstrated and learned in such a way, that the contact forces between the piston and the barrel were minimized. During the execution, higher contact forces arose due to the trajectory tracking errors as a result of the increased velocity of the recorded trajectory.

We have also implemented a safety procedure, which interrupts the trajectory if high contact forces are detected. In order to prevent damage to the robot and other equipment, which might occur at high velocity impacts, the lower limit of the temporal scaling factor ν was set to 0.4. Parameter settings for all experiments in this paper are summarized in Table I. First we tested temporal scaling adaptation using feedback error, as described in Section IV-A. After adaptation, the robot did not succeed to insert the piston into the barrel due to a large trajectory tracking error, which was caused by excessive temporal scaling factor ν learned by the proposed approach. Note that lower gain K resulted only in a slower adaptation, so lowering the gain did not help. In order to accomplish the given task with this approach, we had to additionally limit the temporal scaling factor ν to 0.6.

TABLE I

PARAMETER SETTINGS USED IN ALL EXPERIMENTS

minimal scale factor	ν_{min}	0.4
maximal force	f_{max}	5 N
maximal torque	m_{max}	0.2 Nm
temporal scaling	ξ_ν	10
force scale	ξ_f	1
torque scale	ξ_m	10
P gain	K	0.01
ILC gain	κ	0.98
ILC gain	η_1	0.06
ILC gain	η_2	0.04
exploration noise	σ^2	0.7
discount factor	γ	0.5
number of kernels	N	20
number of kernels	M	20
sampling time	Δt	0.01 s

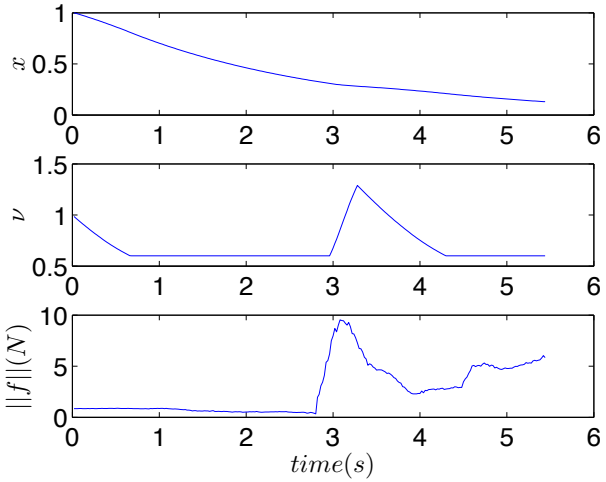

Fig. 3. Phase, temporal scaling factor, and the resulting force norm using feedback error adaptation

Fig. 3 shows that the algorithm increased the velocity in the no-contact zone and reduced it when the contact arose. The total speed-up of the given trajectory was by a factor of 1.47. As discussed earlier, the main problem with this approach is that it can not anticipate impacts, since this would require prediction.

The next experiment was executed under the same conditions using ILC-based approach, as presented in Section IV-B. We performed 5 learning iterations (roll-outs). Results are shown in Fig. 4 As we can see, the ILC approach anticipates the contact force error and can successfully slow down the execution of the task prior to the occurrence of high contact forces. It is worth mentioning that the ILC approach can also learn from unsuccessful attempts, when the trajectory was interrupted due to excessive contact forces. High forces and consequently high errors occurred when the robot has missed to insert the piston into the barrel. This error is than used in the next iteration to modify the temporal scaling factor. Here, a problem arises how to recover the control and the error signal when the trajectory was interrupted. In such a case, the missing part of the control and error signal was recovered from the last successfully executed iteration. The average speed up factor was 1.9 with ILC approach. In summary, the ILC approach performs substantially better comparing to the approach based on feedback error. It learns better temporal scaling profile, can set the proper initial values, and learns in smaller increments, which is safer for both the robot and the environment.

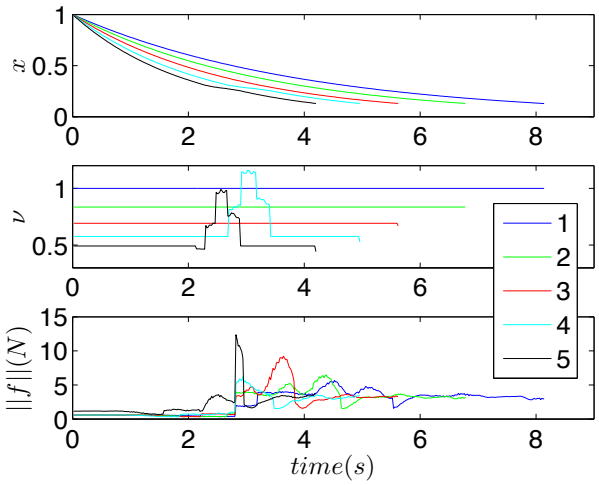

Fig. 4. Phase, temporal scaling factor, and the resulting force norm using the ILC-based approach. Numbers 1-5 in the legend denote the learning iterations

The last experiment was performed using reinforcement learning as described in subsection IV-C. In this case, the only tuning parameter is the exploration noise σ^2. We performed 40 roll-outs of the experiment, each 10th was executed without the exploration noise in order to check the convergence of the learning algorithm. The results are shown in Fig. 5. The average speed up factor was 1.75 with the RL approach, which is lower than with ILC approach. On the other hand, the RL approach generated smoother velocity profile, results in lower force/torque norm and required less tuning, therefore we consider this approach favorable, even though the number of trials was considerably higher than with the ILC-based approach. Fig. 6 shows collected terminal rewards during learning. Red circles denote roll-outs without the exploration noise. Red crosses denote roll-outs, where the execution was interrupted due to the excessive contact forces. Experimental results show that if we continue to explore performing additional roll-outs, more and more attempts will be unsuccessful. This can be explained by the fact that the system has already learned the optimal temporal scaling factor $\nu(x)$ and exploration often results in failure due to excessive scaling.

VI. CONCLUSIONS

In the paper we proposed a new approach to self-adaptation of a given control policy. We focused on speed adaptation. For this purpose, we considered three different approaches: 1) feedback error adaptation, 2) ILC-based approach, and 3) reinforcement learning. We proposed an extension to the reinforcement learning algorithm PoWER,

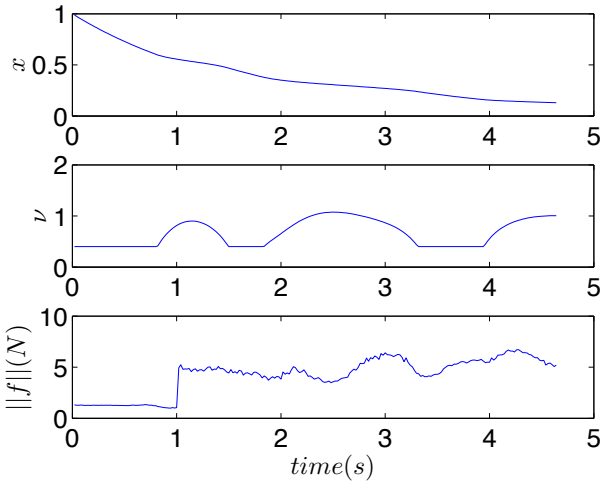

Fig. 5. Phase, learned temporal scaling factor and the resulting force norm using reinforcement learning

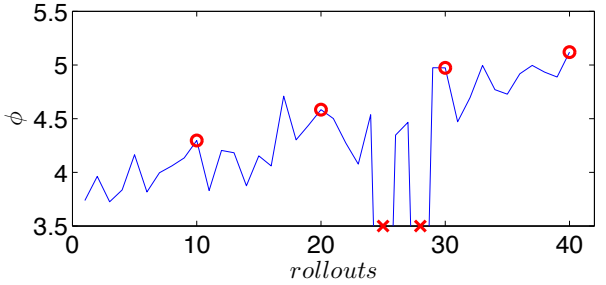

Fig. 6. Terminal rewards collected during reinforcement learning. Red circles denote roll-outs without the exploration noise. Red crosses denote roll-outs, where the execution was interrupted due to the excessive contact forces.

which uses also intermediate rewards for policy updates. The benefit of the proposed algorithm is not only faster learning, but also lower estimated parameter variation. Experimental evaluation has shown that the direct learning of feedback error is less appropriate for optimizing the speed of the control policy. Both Iterative learning controller and reinforcement learning proved to be effective for this purpose. The advantage of reinforcement learning is that it is model-free and does not require many tuning parameters. In our experiments, the initially demonstrated policy was carefully learned in order to be fast and to exhibit low contact forces. Despite of that, the proposed algorithms were able to substantially speed up the task execution. Failures during the experiments occurred mainly due to high contact forces as a result of poorer tracking error at higher velocities.

Note that we allowed the robot to deviate from the originally learned trajectory during task execution. In many cases, some trajectory tracking error is acceptable, e. g. during non-contact approach and depart movements. It is far more important that the task is accomplished quickly and successfully.

REFERENCES

[1] D. M. Wolpert, J. Diedrichsen, and J. R. Flanagan, "Principles of sensorimotor learning." *Nature reviews. Neuroscience*, vol. 12, no. 12, pp. 739–51, Dec. 2011.

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

[3] 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.

[4] F. Stulp, E. Theodorou, and S. Schaal, "Reinforcement learning with sequences of motion primitives for robust manipulation," *IEEE Transactions on Robotics*, vol. 28, no. 6, pp. 1360–1370, 2012.

[5] 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.

[6] E. Burdet, K. P. Tee, I. M. Y. Mareels, T. E. Milner, C.-M. Chew, D. W. Franklin, R. Osu, and M. Kawato, "Stability and motor adaptation in human arm movements," *Biological Cybernetics*, vol. 94, no. 1, pp. 20–32, 2006.

[7] S.-K. Yun, "Compliant manipulation for peg-in-hole: Is passive compliance a key to learn contact motion?" in *IEEE International Conference on Robotics and Automation*, Pasadena, California, 2008, pp. 1647 –1652.

[8] J. Buchli, F. Stulp, E. Theodorou, and S. Schaal, "Learning variable impedance control," *International Journal of Robotics Research*, vol. 30, no. 7, pp. 820–833, 2011.

[9] A. J. Ijspeert, J. Nakanishi, H. Hoffmann, P. Pastor, and S. Schaal, "Dynamical movement primitives: Learning attractor models for motor behaviors," *Neural Computation*, vol. 25, no. 2, pp. 328–373, 2013.

[10] A. Ude, C. G. Atkeson, and M. Riley, "Programming full-body movements for humanoid robots by observation," *Robotics and Autonomous Systems*, vol. 47, no. 2-3, pp. 93–108, 2004.

[11] M. Hersch, F. Guenter, S. Calinon, and A. Billard, "Dynamical system modulation for robot learning via kinesthetic demonstrations," *IEEE Trans. Robotics*, vol. 24, no. 6, pp. 1463–1467, 2008.

[12] B. Nemec, F. Abu-Dakka, J. A. Jørgensen, T. R. Savarimuthu, B. Ridge, J. Jouffroy, N. Krüger, and A. Ude, "Transfer of assembly operations to new workpiece poses by adaptation to the desired force profile," in *IEEE International Conference on Advanced Robots*, Montevideo, Uruguay, 2013.

[13] S. Schaal, P. Mohajerian, and A. Ijspeert, "Dynamics systems vs. optimal control – a unifying view," *Progress in Brain Research*, vol. 165, no. 6, pp. 425–445, 2007.

[14] J. Kober and J. Bagnell, D.and Peters, "Reinforcement learning in robotics: A survey," *International Journal of Robotics Research*, 2013.

[15] 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.

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

[17] B. Nemec, D. Forte, R. Vuga, M. Tamošiūnaitė, F. Wörgötter, and A. Ude, "Applying statistical generalization to determine search direction for reinforcement learning of movement primitives," in *2012 12th IEEE-RAS International Conference on Humanoid Robots*, Osaka, Japan, 2012.

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

[19] 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.

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

[21] K. L. Moore, Y. Chen, and H.-S. Ahn, "Iterative learning control: A tutorial and big picture view," in *Decision and Control, 2006 45th IEEE Conference on*, dec. 2006, pp. 2352 –2357.

[22] D. Bristow, M. Tharayil, and A. Alleyne, "A survey of iterative learning control," *Control Systems, IEEE*, vol. 26, no. 3, pp. 96 – 114, june 2006.

[23] P. Pastor, L. Righetti, M. Kalakrishnan, and S. Schaal, "Online movement adaptation based on previous sensor experiences," in *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, Sept., pp. 365–371.