9th IEEE-RAS International Conference on Humanoid Robots
December 7-10, 2009 Paris, France

Task adaptation through exploration and action sequencing

Bojan Nemec Minija Tamošiūnaitė Florentin Wörgötter Aleš Ude

Abstract— **General-purpose autonomous robots need to have the ability to sequence and adapt the available sensorimotor knowledge, which is often given in the form of movement primitives. In order to solve a given task in situations that were not considered during the initial learning, it is necessary to adapt trajectories contained in the library of primitive motions to new situations. In this paper we explore how to apply reinforcement learning to modify the subgoals of primitive movements involved in the given task. As the underlying sensorimotor representation we selected nonlinear dynamic systems, which provide a powerful machinery for the modification of motion trajectories. We propose a new formulation for dynamic systems, which ensures that consecutive primitive movements can be splined together in a continuous way (up to second order derivatives).**

I. Introduction

Action generation and trajectory modulation are among the most important issues in humanoid robot motor control. An often used paradigm is learning from demonstration or imitation learning [10], where the demonstrated action is used to seed the learning process. Due to different kinematic and dynamic capabilities of the human demonstrator and the target humanoid robot, demonstrated trajectories cannot be simply copied as sequences of joint angles [15], but need to be adapted to the capabilities of the robot. Such problems can be avoided by kinesthetic guiding [3], where the robot arm is led through the action by a human teacher, but this method is not applicable to every robot. Even after the observed trajectories have been made feasible with respect to the robot's kinematics and dynamics, they still need to be modified when the configuration of the external world changes compared to the initial demonstration. Thus a suitable, higher-level adaptation process is needed to change the learned trajectories.

The adaptation can take place in the form of an autonomous exploration, where the robot modifies the available movements by exploring its action space in the neighborhood of the previously acquired movements, thus continuously expanding the available knowledge until optimal (or satisfactory) solution is found. This process is often realized using reinforcement learning techniques. Since we are interested in the development of intelligent robots in household environments, we took the pouring of a liquid into a glass as a representative example in our evaluation experiments. Cup filling is an appropriate task for learning because in general liquid streams are hard to model if one considers arbitrary vessels and liquids. Even if the appropriate robot movements that solve the task for some glasses and liquids are available, these movements require further adaptation procedures if the relative position of the glass with respect to the robot changes or if one of the vessels changes.

When considering reinforcement learning (RL) for attaining a delayed reward, learning at different levels of abstraction is possible. One possibility is to parametrize the shape of the selected trajectory and perform learning by adapting these parameters, so that the overall movement is changed towards better performance. Examples of adaptation of this type are provided in [7], [8], using specific policy gradient technique called natural actor-critic. Another possibility is to leave the parameters specifying the shape of the trajectory as they are. Instead, higher-level parameters specifying the relationship between the movement and the task space can be adapted, e. g. by shifting the trajectory to a new location in 3-D space, this way reducing the problem to an easier reinforcement learning problem with a smaller number of parameters. This second approach was chosen in our research. We applied reinforcement learning with function approximation and continuous actions.

To generate subgoals suitable for this kind of reinforcement learning, it is often useful to segment the overall movement into primitive movements that are related to the subgoals of the task. Such methodology was also used in [2] in the context of learning from demonstration and practicing. In this paper we employ dynamic movement primitives (DMPs) [4], [11], which are essentially parametrized trajectories encoded by dynamic systems, as a basic movement representation. DMPs explicitly contain the final goal position of the primitive motion among the parameters and thus provide suitable higher-level parameters for reinforcement learning. Since a smooth transition between primitive movements without coming to a full stop is often needed to effectively sequence the primitives, we propose a new formulation of dynamic systems that ensures smoothness (up to derivatives of second order) of the transition between two consecutive primitive movements.

In the following we first study movement sequencing with DMPs. In the second part we investigate an application of reinforcement learning to the adaptation of the available motor primitives in the context of cup filling.

II. Sequencing of Motion Primitives

Lets briefly consider the task of pouring a liquid into a glass. It depends on many factors including the position of

B. Nemec and A. Ude are with the Jožef Stefan Institute, Department of Automatics, Biocybernetics, and Robotics, Ljubljana, Slovenia, bojan.nemec@ijs.si, ales.ude@ijs.si

M. Tamošiūnaitė is with the Vytautas Magnus University, Kaunas, Lithuania, m.tamosiunaite@if.vdu.lt

F. Wörgötter is with the Bernstein Center for Computational Neuroscience, Göttingen, Germany, worgott@bccn-goettingen.de

the glass with respect to the body, the shape of the vessel containing the liquid and the shape of the glass to be filled. A general strategy for pouring is 1) approach the glass to be filled with a suitable approach trajectory and 2) start the pouring motion towards the end of the approach trajectory and execute the pouring motion while controlling the liquid flow until the glass is filled to the desired level. These two phases define two separate movement primitives. Note that there is a smooth transition between the two phases, i. e. the hand motion does not come to a full stop while transitioning from the approach phase to the pouring phase.

To successfully fill the glass placed at different locations on the table, the actual pouring motion does not need to be changed. Successful pouring can be achieved solely by selecting the appropriate goal position for the approach trajectory, which is automatically taken into consideration by the underlying dynamic system, followed by the previously learned and constant pouring movement. The goal position of the approach trajectory provides the parameters that can be learned by reinforcement learning. For this approach to work, we need to be able to smoothly sequence the available primitives. In this section we propose a dynamic systems formulation that allows smooth sequencing of DMPs without coming to a full stop at the end of each movement, as it is necessary in the case of pouring movements. While some of these parameters could be inferred analytically, this is at least a non-trivial task because many parameters need to be considered (shape of the glass, properties of liquid flow for different liquids, capabilities of the robotic arm, ...).

In the standard DMP formulation for discrete movements, motion in each task coordinate is represented as a damped mass-spring system perturbed by an external force. Such a system can be modeled with a set of differential equations [11][1]

$$\begin{aligned} \dot{v} &= \frac{1}{\tau}\left(K(g-y) - Dv + f(x)\right), \\ \dot{y} &= \frac{v}{\tau} \end{aligned} \tag{1}$$

where v and y are the velocity and position of the system, x is the phase variable, which defines the time evolution of the trajectory, τ is the temporal scaling factor, K is the spring constant, and D is the damping. The phase variable x is defined by

$$\dot{x} = -\frac{\alpha x}{\tau}. \tag{2}$$

For trajectory generation it is necessary that the dynamic system is critically damped and thus reaches the goal position without overshoots. A suitable choice is $D = 2\sqrt{K}$, where K is chosen to meet the desired velocity response of the system. Function $f(x)$ is a nonlinear function which is used to adapt the response of the dynamic system to an arbitrary complex movement. A suitable choice for $f(x)$ was proposed by Ijspeert et al. [4] in the form of a linear combination of M radial basis functions

$$f(x) = \frac{\sum_{j=1}^{M} w_j \psi_j(x)}{\sum_{j=1}^{M} \psi_j(x)} x, \tag{3}$$

where ψ_j are Gaussian functions defined as $\psi_j(x) = \exp(-\frac{1}{2\sigma_j^2}(x - c_j)^2)$. Parameters c_j and σ_j define the center and the width of the j-th basis function, while w_j are the adjustable weights used to obtain the desired shape of the trajectory.

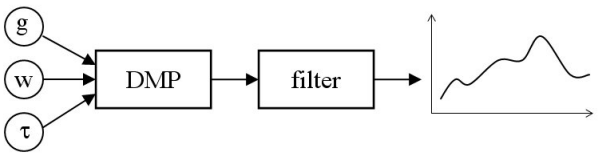

Fig. 1. Filtered output of the DMP

In the original DMP formulation [4], the system has an initial state $(x, y, v) = (1, y_0, 0)$ and a final state $(x, y, v) = (0, g, 0)$, which means that the previous motion has to completely stop before the next motion is generated. Pastor et al. [6] noted that by appropriately defining the initial conditions for the second movement, two consecutive movements can be joined together with continuous velocities. The acceleration, however, remains discontinuous even after this modification. Here we eliminate this restriction by replacing the second order system (1) with a third order system.

In order to overcome the jumps in velocities and accelerations when joining two trajectories, we propose to apply a first order low-pass filter at the output of the DMP generator, as shown in Fig. 1. The second order system (1) now turns into a third order system, which is defined by equations

$$\begin{aligned} \dot{v} &= \frac{1}{\tau}\left(K(g-y) - Dv + f(x)\right), \\ \dot{y} &= \frac{v}{\tau}, \\ \dot{q} &= \frac{H}{\tau}(y - q). \end{aligned} \tag{4}$$

Here H is an appropriately chosen filter constant and q is the new output of the modified DMP. The phase variable x remains defined by Eq. (2). Like in the original formulation (1), the third order system is stable if the constants K, D, H, and τ are appropriately selected. The proof is as follows. It is easy to see that if we omit the nonlinear term f from Eq. (4), a general solution for the remaining linear differential equations system is given by

$$\begin{bmatrix} v \\ y \\ q \end{bmatrix} = \begin{bmatrix} 0 \\ g \\ g \end{bmatrix} + \exp(t\mathbf{A})\,\mathbf{c},\ \mathbf{A} = \begin{bmatrix} -\frac{D}{\tau} & -\frac{K}{\tau} & 0 \\ \frac{1}{\tau} & 0 & 0 \\ 0 & \frac{H}{\tau} & -\frac{H}{\tau} \end{bmatrix}, \tag{5}$$

where $\mathbf{c} \in \mathbb{R}^3$ is an arbitrary constant, which is determined from the initial conditions. The system is guaranteed to

[1]While constants are denoted differently in this paper, the two formulations are equivalent.

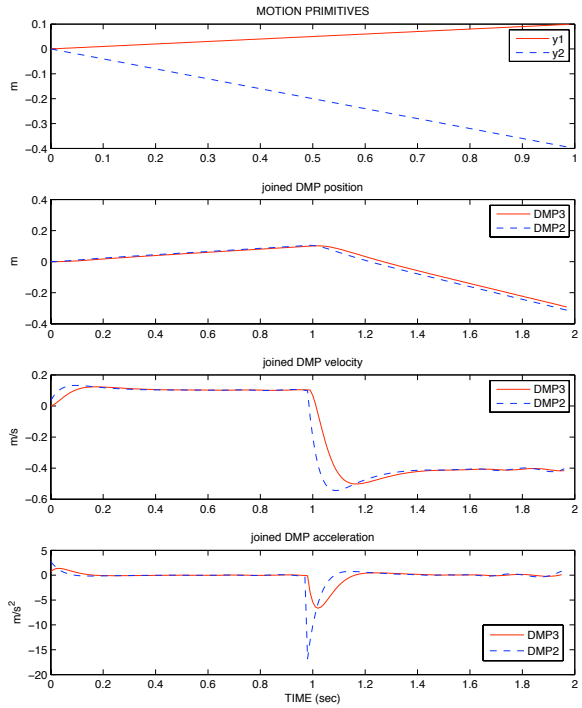

Fig. 2. Sequencing of dynamic movement primitives: the output of the second and third order DMP

converge to the attractor point $[0, g, g]$ if the eigenvalues of $\mathbf{A}$ are negative. The eigenvalues of $\mathbf{A}$ are given as solutions to the equation

$$\det(\mathbf{A}-\lambda\mathbf{I}) = -\left(\lambda^2 + \lambda D/\tau + K/\tau^2\right)(H/\tau + \lambda) = 0, \quad (6)$$

thus $\mathbf{A}$ has negative eigenvalues $-\sqrt{K}/\tau$ and $-H/\tau$ if $D = 2\sqrt{K}$, $H, K, \tau > 0$. Since the phase x and consequently the nonlinear term $f(x)$ tend to zero, the nonlinear system (4) is also guaranteed to converge to the attractor point $[0, g, g]$.

In simulation we have tested the sequencing of linear movements encoded by DMPs. Figure 2 shows the response of two consecutive second and third order DMPs and the corresponding velocities and accelerations. As we can see, the modified DMP formulation ensures continuous accelerations, whereas the accelerations in the original formulation are discontinuous.

A. Motion Acquisition

The trajectory represented by a third order dynamic system is parameterized with the initial acceleration, velocity, and position, the final goal position, and a set of weights w_j associated with radial basis functions. In this section we present the procedure for the calculation of weights w_j. We assume that from human demonstration or kinesthetic guiding we obtain trajectory data points $\{q_d(t_i), \dot{q}_d(t_i), \ddot{q}_d(t_i), \dddot{q}_d(t_i)\}_i$, $t_i \in [0, T]$. Note that unlike in the original DMP formulation, this formulation requires

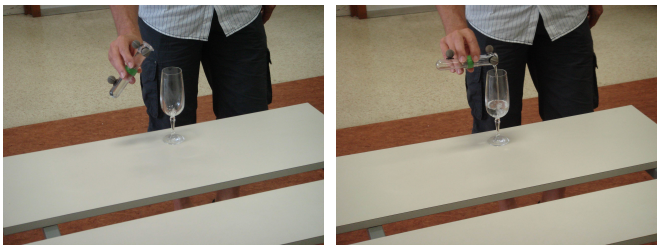

Fig. 3. Two frames from the pouring demonstration. The motion was captured by attaching markers to the container.

also to estimate the jerk. We define function f^* as follows

$$f^*(t) = \frac{\tau^3}{H}\dddot{q} + \tau^2\left(1 + \frac{D}{H}\right)\ddot{q} + \tau\left(\frac{K}{H} + D\right)\dot{q} + K(q-g), \quad (7)$$

where $q = q(t)$. This function is obtained by replacing the system of three first order equations (4) with one equation of the third order, where the nonlinear term $f(x)$ has been omitted. Our task is to find a set of weights $\{w_j\}$ that minimize the quadratic cost function

$$J = \sum_{i=0}^{N}(f^*(t_i) - f(x(t_i)))^2. \quad (8)$$

We use global regression methods to find the optimal weights w_j. Other authors [4], [6] applied locally weighted regression, which instead minimizes M separate cost functions

$$J_j = \sum_{i=0}^{N}\psi_j(x(t_i))(f^*(t_i) - w_j x(t_i))^2, \quad (9)$$

$j = 1, \ldots, M$. Locally weighted regression [1] was proposed as a method that prevents negative interference between task models. Local models are used to generalize in the neighborhood of the given data point. However, in the context of trajectory generation from human demonstration, a complete trajectory is observed over the entire time interval, therefore locally weighted regression has no advantages over the global regression except for the lower computational burden. On the other hand, when using global regression, significantly less kernel functions are necessary to encode the trajectory within the required precision and consequently less computation is required to track the previously calculated trajectory. Global regression results in the following linear system of equations

$$\mathbf{A}\mathbf{w} = \mathbf{f}^*, \quad (10)$$

$$\mathbf{w} = \begin{bmatrix} w_1 \\ \vdots \\ w_M \end{bmatrix}, \quad \mathbf{f}^* = \begin{bmatrix} f^*(t_0) \\ \vdots \\ f^*(t_N) \end{bmatrix},$$

$$\mathbf{A} = \begin{bmatrix} \frac{\psi_1(x(t_0))x(t_0)}{\sum_{j=1}^{M}\psi_j(x(t_0))} & \cdots & \frac{\psi_M(x(t_0))x(t_0)}{\sum_{j=1}^{M}\psi_j(x(t_0))} \\ \vdots & \vdots & \vdots \\ \frac{\psi_1(x(t_N))x(t_N)}{\sum_{j=1}^{M}\psi_j(x(t_N))} & \cdots & \frac{\psi_M(x(t_N))x(t_N)}{\sum_{j=1}^{M}\psi_j(x(t_N))} \end{bmatrix}.$$

Similarly as in the case of locally weighted regression, it is possible to compute a solution to (10) recursively by incrementally updating the following quantities

$$\mathbf{P}_i = \mathbf{P}_{i-1} - \frac{\mathbf{P}_{i-1}\mathbf{a}_i\mathbf{a}_i^T\mathbf{P}_{i-1}}{1+\mathbf{a}_i^T\mathbf{P}_{i-1}\mathbf{a}_i}, \quad (11)$$

$$\mathbf{w}_i = \mathbf{w}_{i-1} + (f^*(t_i) - \mathbf{a}_i^T\mathbf{w}_{i-1})\mathbf{P}_i\mathbf{a}_i, \quad (12)$$

where $\mathbf{a}_i$ is the M dimensional column vector associated with the corresponding row of the matrix $\mathbf{A}$ and the optimal weights are $\mathbf{w} = \mathbf{w}_N$.

Using the proposed estimation method, we captured and successfully reconstructed the approach and the pouring movements (see Fig. 3). An optical tracking system with passive markers attached to the container was utilized to capture the motion of the container. The motion of the container was then mapped to the 3-D space motion of the robotic hand.

III. Goal Configuration Adaptation Using Reinforcement Learning

Now we turn our attention towards the adaptation of the available action knowledge through exploration. In the interest of better understanding of the learning process, we fist describe our experimental setup. The implemented learning process is, however, more general and is not limited to this experiment.

A. Experimental setting

The evaluation experiments were performed using two robots; a humanoid robot HOAP-3 and a seven degrees of freedom (DOFs) robotic arm PA-10. HOAP-3 is a small humanoid robot, whose arm has a limited workspace (5 DOFs), while the seven degress of freedom available to PA-10 allow the arm to reach any desired position and orientation of the wrist within the robot workspace.

The execution of the pouring action consists of two phases; the approach movement and the actual pouring movement. The pouring movement of HOAP-3 was fixed and was not changed during the exploration process. What the robot needed to learn was the optimal position and orientation of the robot's hand from where to start the pouring movement

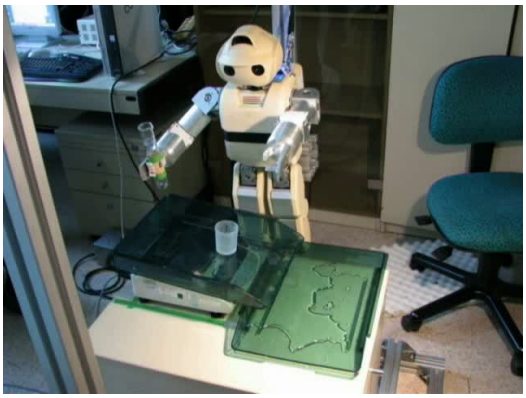

Fig. 4. Experimental setup with HOAP-3 humanoid robot

(with respect to the location of the glass to be filled). This parameter is given as a final position on the approach trajectory and is explicitly encoded within a DMP. Obviously, the optimal starting position for pouring changes when the glass is moved to a new location. However, because the arm has only 5 degrees of freedom and because of nonlinearity of the robot's kinematics, the new optimal position is not a linear function of the glass location. The task space to be explored was further limited by specifying the height from which to start pouring, which was defined as a function of the distance of the glass from the robot body using the formula

$$z = 0.5((x-0.1) + (-0.14-y)) + 0.01, \quad (13)$$

where (x, y, z) are the robot-centered coordinates with the following directions (x - forward, y - to the side, z - up). Thus the wrist was kept lower when being close to the body, and higher going away from the body. This was important to move the arm at sensible orientations because the 5 degrees of freedom arm of HOAP-3 cannot reach every desired configuration in the 6-D task space. The 3-D wrist orientation was defined so that the glass was not tilted at the beginning of the pouring movement, which fixed the remaining two degrees of freedom.

The reinforcement learning process described in the next sections explored the so defined planar surface to find the optimal position for pouring with respect to the given glass location. The outcome of pouring, which provided the reward for reinforcement learning, was measured using a scale that weighted the amount of liquid that remained in the glass to be filled (see Fig. 4). The scale could be replaced by a vision system measuring the level of liquid in the glass, but this was not the focus of our investigation.

We also evaluated our algorithms with a PA-10 robot arm. Since the problem to keep the wrist at a specific angle does not arise when using a 7 degrees of freedom arm, we now allowed variable height in addition to the y coordinate (direction to the side of the robot), but x coordinate (forward) was kept fixed. This coordinate can be estimated by vision. The 3-D wrist orientation was defined so that the glass was not tilted at the beginning of the pouring motion and the wrist axis orientation was parallel to the robot's sagittal plane.

B. Exploration

We implemented a reinforcement learning method with function approximation. We define the value function $V(s)$ as follows

$$V(s) = \sum_{k=1}^{N} \theta^k \Phi^k(s) / \sum_{k=1}^{N} \Phi^k(s) \quad (14)$$

where $\Phi^k(s)$ is the activation function of the k-th kernel in state s, θ^k are the weights associated with the k-th kernel function, and N is the overall number of kernels in the system. In the context of the our experiment, state s is defined as a pair of Cartesian coordinates at which the robot starts pouring the liquid ($[x, y]^T$ and $[y, z]^T$ for HOAP-3 and PA-10, respectively). Weights θ are adapted through learning

as described in the next section. We used spherical Gaussian kernels, uniformly distributed over the analyzed area with $\sigma = 0.5\ cm$ for the HOAP-3 setup and $\sigma = 2.1\ cm$ for the PA-10 setup. $N = 2000$ was used in both cases to thoroughly exclude effects from insufficient coverage of the state space, although according to our experience with function approximation methods of similar complexity [13], ten times smaller amount of kernels would often suffice to adequately represent the investigated space.

To define a new exploratory action, gradient of the current estimate of the value function was calculated $\Delta = \mathrm{grad}\, V(s)$, and an action was performed taking the direction of the gradient into account. Instead of using pure gradient ascent, the update for the next state was calculated as a combination of the current gradient and the previous action $A_{previous}$:

$$\Delta_{current} = (\Delta x, \Delta y)_{current}, \tag{15}$$

$$A_{previous} = (Ax, Ay)_{previous}, \tag{16}$$

which resulted in

$$A_{final} = c\Delta_{current} + (1 - c)A_{previous}, \tag{17}$$

where at the beginning of learning $c = 1.5$ was used, while in later trials c was reduced by 0.1 trial by trial. The proposed smoothing procedure helps to avoid jerky exploratory movements in the beginning of the learning process and also to some degree influences the process of refinement in RL with function approximation. Traditional proofs of convergence for RL are no longer valid when using action smoothing, but in practice learning converged reliably.

In the beginning of learning, the probability of random exploration was set to 0.5 due to the fact that the chosen method for performing continuous actions in the value function approximation scheme was sometimes attracted towards local minima. The exploration was diminished by 0.05 in each trial to adhere to the learned components better.

C. Learning method

In the proposed learning framework, the change in the value of θ^k follows the mean across all activated kernels of the next state s'

$$\theta^k = \theta^k + \mu[r + \gamma V(s') - \theta^k]\Phi_N^k(s), \tag{18}$$

where k is the number of the kernel to which the weight is associated, r is the reward, $\mu < 1$ the learning rate, $\gamma < 1$ the discount factor, $V(s')$ is the value of the next state, and $\Phi_N^k(s)$ is the normalized activity function for kernel k in state s

$$\Phi_N^k(s) = \Phi^k(s) / \sum_{k=1}^{N} \Phi^k(s). \tag{19}$$

The rule (18) is called averaging function approximation rule and is considered to perform more stably [9] in function approximation schemes as compared to standard methods (e. g. see [12]).

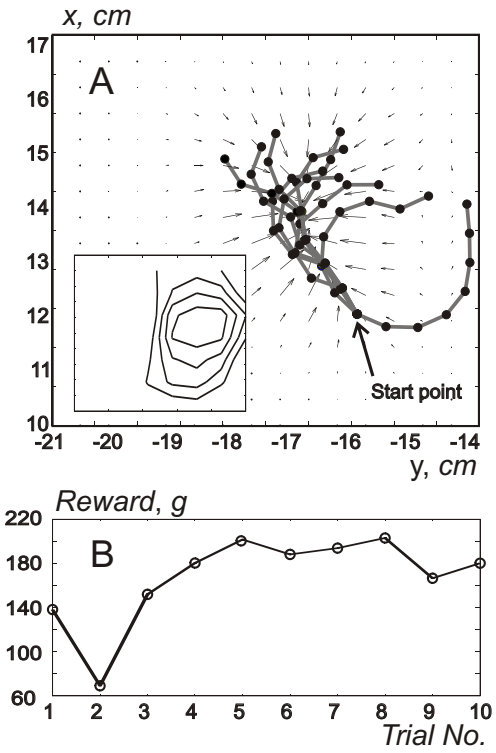

Fig. 5. Learning trajectories, vector field obtained in the process of learning (marked with pointers), and the contour plot of the reward (succesful pouring amounts) obtained through sampling (A); amount of reward accumulated in successive trials (B). Real HOAP-3 robot was used in this case.

D. Results

In experiments with HOAP-3, ten trials with eight pouring attempts each were performed. Exploratory paths together with the resultant vector field are shown in Fig. 5.A. All trials started at the same initial hand position and orientation. One can see that the learning was successful and the rewards starting with the 3rd trial were always high (see Fig. 5.B). Some oscilations in the reward profile are due to the random exploration component.

Several improvements were introduced to make learning quicker: 1) restarting the next trial from the best point of the previous trial; 2) shortening the step size with the number of trials; and 3) reducing the amount of smoothing of the trajectory with the number of trials. This led to a more reliable learning both in simulation and on the robot. The exploratory paths at the end of learning together with the obtained vector field are shown in Fig. 6.A. Statistics plots are shown in Fig. 6.B.

A similar experiment was performed using the PA-10 robot arm. Examples of the exploratory paths are shown in Fig. 7. In Fig. 7.A, a few initial paths are shown, while in Fig. 7.B, exploratory paths at the end of learning and the obtained vector field, are shown. The inset of Fig. 7.A shows the reward contours obtained through sampling. One can see from the inset that the same reward is obtained independently of the wrist height (z coordinate). Learning chooses an arbitrary height but prefers the appropriate y coordinate. Vector field

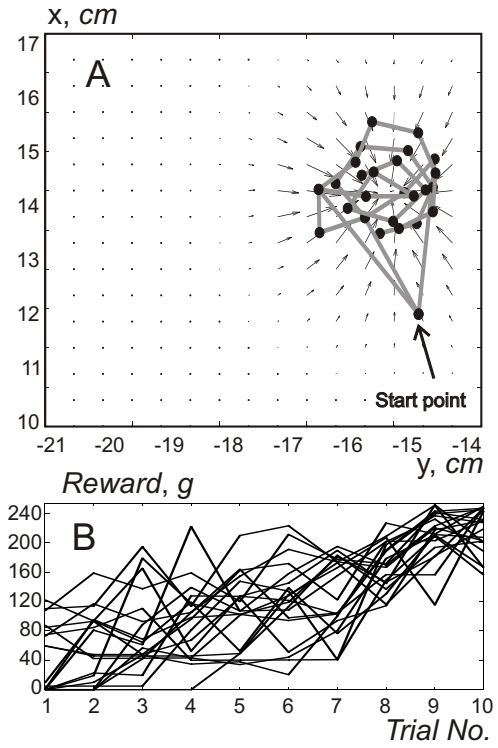

Fig. 6. Exploratory paths of the hand traversed during learning and the vector field obtained at the end of learning when using the strategy of jumping directly to the best point of the previous trial (A); sequences of accumulated reward for 20 learning experiments obtained using HOAP-3 simulator (B).

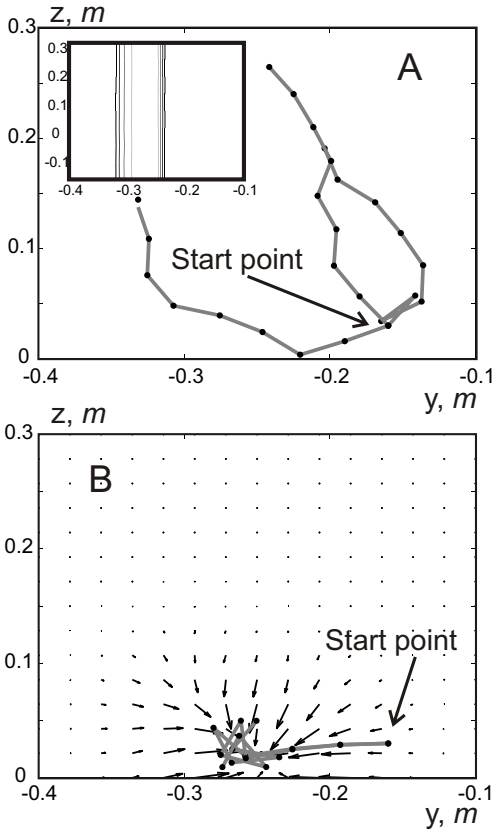

Fig. 7. Exploratory paths obtained at the beginning of learning in the experiment with PA-10 (A). The inset shows contours of the reward area obtained using sampling. Exploratory paths and vector field obtained at the end of learning are shown in (B).

is also stronger along the y coordinate as compared to the height coordinate.

In PA-10 experiments, good performance was obtained after 6-8 trials. In Fig. 8.A accumulated rewards over sequential trials from 10 experiments are shown. The average over those experiments is given in Fig. 8.B. In the current setup with PA-10, learning effects were achieved several trials later as compared to the analyzed HOAP-3 setup, which is probably due to the bigger area that needs to be explored.

IV. Summary and Conclusion

A reinforcement learning procedure with function approximation and continuous actions, which are encoded by dynamic systems, was developed. To support action sequencing, a new formulation for dynamic systems was proposed. This methodology was successfully applied to learn appropriate robot movements for liquid pouring. We have shown that good performance can be achieved within 3-8 trials (exploratory paths, or 20 to 60 attempts to pour), on two different robot arms. The number of trials is acceptable for a robotic application, where procedures with thousands of learning trials are often not feasible.

There are alternative ways to refine robot movements using reinforcement learning. In [8], a novel policy gradient method called *natural actor-critic* was used to solve the problem of throwing a ball into a cup with a 7 degrees of freedom robot arm. The method exhibits excellent performance when applied to problems involving multiple dimensions. One problem we had with the natural actor-critic technique – at least in our experiments – was that the initial trajectory had to be relatively close to the desired trajectory if one wants to achieve efficient convergence [14]. In comparison, approaches like ours need to explore a smaller task space to find a satisfactory solution and are therefore less susceptible to a bad initial approximation. Also, value function based reinforcement learning methods deal with delayed rewards in a systematic way, whereas gradient based methods like the natural actor-critic would fail because of the flat reward surface. This is another reason for why our method can correct larger inaccuracies in the demonstration as compared to the natural actor-critic. Hence although with methods like natural actor-critic, reinforcement learning can be applied to large state spaces, an alternative to divide the task into several smaller problems and solve one or several of the smaller tasks with reinforcement learning may provide more stable results. While other systems in which the task has been subdivided into smaller tasks by defining suitable subgoals were developed in the past [5], our approach shows how to refine the available movement primitives by encoding them as dynamic systems. We demonstrated that dynamic systems provide a suitable framework for task decomposition and proposed a formulation that allows smooth sequencing of

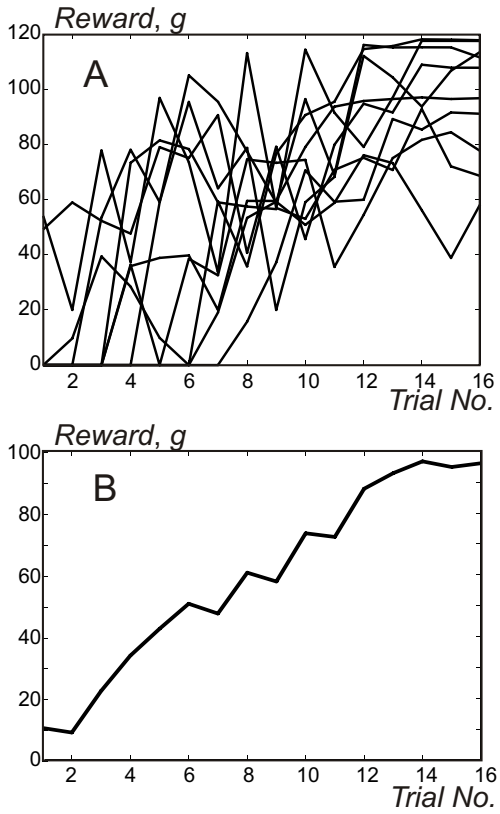

Fig. 8. Ten sequences of accumulated reward in PA-10 experiments (A); average value of the ten reward sequences (B).

consecutive movement primitives up to second-order derivatives.

Acknowledgment: The work described in this paper was conducted within the EU Cognitive Systems project PACO-PLUS (FP6-2004-IST-4-027657) funded by the European Commission.

REFERENCES

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

[2] D. C. Bentivegna, C. G. Atkeson, A. Ude, and G. Cheng, "Learning to act from observation and practice," *International Journal of Humanoid Robotics*, vol. 1, no. 4, pp. 585–611, 2004.

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

[4] 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.

[5] J. Morimoto and K. Doya, "Acquisition of stand-up behavior by real robot using hierarchical reinforcement learning." *Robotics and Autonomous Systems*, vol. 36, pp. 37–51, 2001.

[6] P. Pastor, H. Hoffmann, T. Asfour, and S. Schaal, "Learning and generalization of motor skills by learning from demonstration," in *Proc. IEEE Int. Conf. Robotics and Automation*, Kobe, Japan, 2009, pp. 763–768.

[7] J. Peters and S. Schaal, "Reinforcement learning for parameterized motor primitives," in *International Joint Conference on Neural Networks*, 2006, pp. 73–80.

[8] ——, "Reinforcement learning of motor skills with policy gradients." *Neural Networks*, vol. 21, pp. 682–697, 2008.

[9] S. I. Reynolds, "The stability of general discounted reinforcement learning with linear function approximation." in *UK Workshop on Computational Intelligence (UKCI-02)*, September 2002, pp. 139–146.

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

[11] 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.

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

[13] M. Tamosiunaite, J. Ainge, T. Kulvicius, B. Porr, P. Dudchenko, and F. Wörgötter, "Path-finding in real and simulated rats: assessing the influence of path characteristics on navigation learning." *Journal of Computational Neuroscience*, vol. 25, pp. 562–582, 2008.

[14] M. Tamosiunaite, T. Asfour, and F. Wörgötter, "Learning to reach by reinforcement learning using a receptive field based function approximation approach with continuous actions." *Biological Cybernetics*, vol. 100, no. 3, pp. 249–260, 2009.

[15] 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.