Exploration in Structured Space of Robot Movements for Autonomous Augmentation of Action Knowledge

Denis Forte, Bojan Nemec and Aleš Ude
Humanoid and Cognitive Robotics Lab
Department of Automatics, Biocybernetics and Robotics
Jožef Stefan Institute, Ljubljana, Slovenia
Email: denis.forte@ijs.si, bojan.nemec@ijs.si, ales.ude@ijs.si

***Abstract*—Imitation learning has been proposed as the basis for fast and efficient acquisition of new sensorimotor behaviors. Movement representations such as dynamic movement primitives were designed to enable the reproduction of the demonstrated behaviors and their modulation with respect to unexpected external perturbations. Various statistical methods were developed to generalize the acquired sensorimotor knowledge to new configurations of the robot's workspace. However, statistical methods can only be successful if enough training data are available. If this is not the case, usually the teacher must provide additional demonstrations to augment the database, thereby improving the performance of generalization. In this paper we propose an approach that enables robots to expand their knowledge database autonomously. Efficient exploration becomes possible by exploiting the structure of the search space defined by the previously acquired example movements. We show in real-world experiments that this way the robot can expand its database and improve the performance of generalization without the help of the teacher.**

***Keywords*—imitation learning, reinforcement learning, dynamic movement primitives, statistical generalization, training data, autonomous database expansion.**

I. Introduction

It has been suggested that imitation learning can provide means to limit the huge state-action space of high degree of freedom robots such as humanoids [1]. Through imitation a robot can gain the first approximation of the desired movement, which can later be improved by autonomous exploration. It has been demonstrated that this way the robot can acquire difficult to learn behaviors, e. g. the game of kendama, which consists of two parts, throwing a ball up on an attached string and catching it in a cup [2]. Based on these early works, reinforcement learning of robot movements has started to be seen as a viable approach to motion learning in robotics [3]–[7], even in the case of high degree freedom humanoid robots. Reinforcement learning algorithms such as PoWER (Policy learning by Weighing Exploration with the Returns) [8] and PI^2 (Policy Improvement with Path Integrals) [5] are able to deal with sensorimotor learning in high dimensional spaces. Reinforcement learning focuses on how to acquire optimal task performance in a given situation. Other researchers have studied how to generalize the available movement primitives to new situations [9]–[12] or represent multiple movements within a dynamical system that can represent multiple control policies [13], [14]. The main difference between both approaches is that in the first approach, the primitives are combined on-line so that the optimal movement for the current situation is generated, whereas the second approach generates a representation of all movements off-line using a global optimization approach.

978-1-4673-7509-2/15/$31.00 ©2015 European Union

The described approaches enable the robot to autonomously improve single movements using reinforcement learning and to generate new variants of the learned behaviors using statistical techniques. In this paper we focus on how the robot can autonomously acquire new example movements for its database, which is the key to improving the performance of the initially roughly learned behavior. Up to now generalization mainly relied on the availability of the sufficient amount of training data, which were provided by the demonstrator. In this paper we show that firstly, the robot can exploit the structure provided by the previously acquired example movements to accelerate its learning process and secondly, that by adding the newly acquired trajectories to the database of example movements we can increase the performance of statistical generalization. Note that successfully solving the first problem does not necessarily mean the accuracy of generalization will also be improved; statistical learning can only be successful if we can ensure that the newly acquired data is correlated with the existing pattern of motor primitives in the database. Thus we must ensure that the new trajectories identified by reinforcement learning are correlated with the ones that are already in the database.

First we obtain a few training movements that solves the given task in some specific situations. Then we apply statistical generalization to compute relatively good initial approximation of new situation inside learning space. As the next step the reinforcement learning is used to refine the approximation in few steps so the robot can accomplish the task correctly. Every learned movement is then stored in the training base, so that additional approximations of different situations can be estimated more accurately and that reinforcement learning can get faster results. When the learning space is fairly revealed, which means that enough training data is acquired, the statistical generalization itself offers movement that is good enough and the reinforcement learning is not needed any more.

II. Acquisition of Example Database and Generalization

We use kinesthetic guiding [15] to acquire the initial example movements. Lets assume that a set of S example trajectories $\mathbf{M}_i,\ i = 1, \dots, S$, has been collected by kinesthetic guiding, and that all these trajectories result in a successful execution of the desired task in different (but related) situations. See Fig. 1 for an example where a human demonstrator guided the robot to pour water from a bottle into a glass located at different positions on the table and with different quantities of water in the bottle across different demonstrations. We call the vector defined by parameters describing the task a query point (in the above example, the query point consists of position of the glass on the table and volume of the water in the bottle). Let $\mathbf{q}_i \in \mathbb{R}^m,\ i = 1, \dots, S$, be the query points associated with every demonstration, where m is the dimensionality of these parameters. The example trajectories $\mathbf{M}_i$ are represented as sequences of positions, velocities and accelerations $\{\mathbf{y}_{ij}, \dot{\mathbf{y}}_{ij}, \ddot{\mathbf{y}}_{ij} \in \mathbb{R}^{dof}\}$, measured at times $t_{ij},\ j = 1, \dots, n_i,\ t_{i1} = 0$, where n_i defines the number of data points on the example trajectory $\mathbf{M}_i$ and dof denotes the number of degrees of freedom on the trajectory. Both Cartesian space and joint space trajectories can be used to define the example data set. In experiments described in this paper we use joint space trajectories.

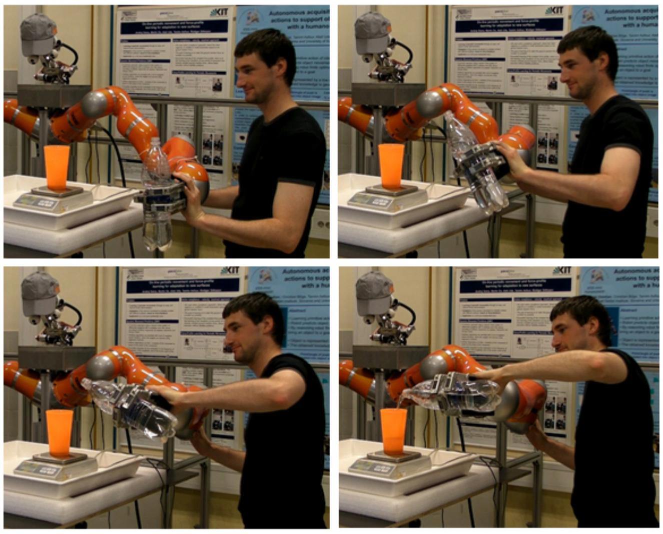

Fig. 1. Acqusition of example movements by kinesthetic guiding.

The aim of generalization is to compute a trajectory that solves the desired task at any given query point $\mathbf{q}$. In the above example, the output trajectory is the appropriate pouring movement. We implemented two different approaches to statistical generalization [9], [11]. In this paper we use the more efficient but less accurate approach described in [11]. This approach is based on dynamic movement primitives (DMPs) [16] and Gaussian process regression (GPR) [17]. For every query point $\mathbf{q}$, GPR computes an approximation of the task solution

$$\mathbf{G}_{\{\mathbf{M}_i,\mathbf{q}_i\}_{i=1}^S} : \mathbf{q} \mapsto \begin{bmatrix} \mathbf{w} \\ \tau \\ \mathbf{g} \end{bmatrix}, \tag{1}$$

where $\mathbf{w}$, τ, and $\mathbf{g}$ are the parameters defining a DMP that describes the movement for the given query point $\mathbf{q}$. For every degree of freedom, the DMP is defined by a second order dynamical system consisting of a linear and nonlinear part, where the linear part has a well-defined attractor dynamics, which ensures the convergence of the system, while the nonlinear part provides the ability to follow any desired trajectory on the given time (phase) interval

$$\begin{aligned} \tau\dot{z} &= \alpha_z(\beta_z(g - y) - z) + f(x), & (2)\\ \tau\dot{y} &= z, & (3)\\ \tau\dot{x} &= -\alpha_x x. & (4) \end{aligned}$$

Here y is the one of the robot's degrees of freedom, z is an auxiliary variable, and x is the phase variable common across all of the robot's degrees of freedom. α_x, α_z, and β_z are constants, which are specified so that the linear part of system (2) – (4) converges to the unique equilibrium point, which is at $z = 0$, $y = g$ and $x = 0$, regardless of where the movement has started. $\tau > 0$ is the temporal scaling factor that can be used to modulate the velocity of the movement. The nonlinear part f of the system contains free parameters w_k that enable the robot to track any desired trajectory from the initial position $(y_0, \dot{y}_0)$ to the final equilibrium point $(0, g, 0)$

$$f(x) = \frac{\sum_{k=1}^N w_k \Psi_k(x)}{\sum_{k=1}^N \Psi_k(x)} x,\ \ \Psi_k(x) = \exp\left(-h_k\left(x - c_k\right)^2\right). \tag{5}$$

Here c_k are the centers of radial basis functions distributed along the trajectory and $h_k > 0$. They can be computed as described in [9]. While α_x, α_z, β_z, τ, and x are the same for all of the degrees of freedom, z, y, g and the parameters w_k vary across the degrees of freedom. Note that since x converges to 0, $f(x)$ also converges to 0 and the system (2) – (4) becomes linear as time tends to infinity (and consequently phase x to zero). Given a new query point $\mathbf{q}$, the algorithm of Forte et al. [11] computes the DMP parameters for all degrees of freedom. Thus $\mathbf{w} \in \mathbb{R}^{N \times dof}$ contains the weights w_k associated with every degree of freedom and vector $\mathbf{g} \in \mathbb{R}^{dof}$ the final configuration of every degree of freedom. τ is associated with the phase variable and is therefore common across all of the degrees of freedom.

The details of the described generalization approach can be found in [11]. What is important for this paper is that for every query point $\mathbf{q}$, the generalization function (1) computes the corresponding DMP. Thus statistical generalization provides a low-dimensional parametrization of the solution space, which can be exploited to accelerate the acquisition of new example trajectories.

III. Expanding the Example Database by Autonomous Exploration

Although generalization can provide an estimate for the trajectory (with respect to the demonstrated ones), which can be used to accomplish the desired task, the computed movement can of course only be good enough for successful task accomplishment if enough training data are available. Additional training examples need to be provided if the task performance is not as good as expected. While a human teacher could provide more data, it would be advantageous if the robot could explore the solution space on its own and expand the database autonomously. In this section we show how to realize such autonomous exploration.

A straightforward approach to generate more data would be to start with the initial approximation for the desired movement as provided by statistical generalization at the given query point and continue with one of the standard reinforcement learning algorithms to find a better solution. There are several problems with this straightforward approach:

- General reinforcement learning is model-free and it might take a long time before a better solution can be found.
- The solution is often not unique, e. g. a given amount of water can be poured into a glass in many different ways, therefore general reinforcement learning might find a solution which is different from the solution selected by the demonstrator. Significantly different movements are not suitable for statistical generalization.

To address the first problem, we exploit the parametrization provided by the statistical generalization function (1). To address the second problem we need to bias the search algorithm towards the trajectory manifold defined by the generalization function.

Another issue to be considered when augmenting the database is that the performance of statistical generalization is usually better if query points are uniformly distributed throughout the desired query point space. In the proposed system, once the robot determines that the accuracy of generalization is insufficient, we systematically add new, uniformly distributed queries to the database and determine the associated movements using the structured reinforcement learning approach.

A. Exploiting the Structure of the Trajectory Space

Given a new query point, the generalization function (1) can calculate the appropriate DMP. To improve the performance of the generalization function, we propose to add additional movements into the existing database of movements, where additional movcements are generated by reinforcement learning. By exploiting the available data, we can organize reinforcement learning around two different types of parameters: query point $\mathbf{q}$, which dimensionality is always low, and DMP parameters ($\mathbf{w}$, τ and $\mathbf{g}$), which combined dimensionality is high. As noted in [18], many tasks require high accuracy only on a low-dimensional manifold of the complete movement space. In all our practical examples, the dimensionality of the query points was between 1 and 3.

We propose to compute the optimal movement at a new query point in two steps. First we perform the search for an optimal query point, from which the DMP parameters are computed using Eq. (1). Since this exploration process is limited to the trajectory manifold defined by (1), the obtained DMP parameters can be further improved by exploration in the full DMP parameter space. We considered two reinforcement learning algorithms to implement these steps: PoWER [19]–[21] which operates using terminal reward only, and PI^2 [5], [22], [23], which can also take into account intermediate rewards. Intermediate rewards are important to ensure that the shape of the new trajectory remains similar to the shape of initial trajectories (see Section III-B).

Thus to learn the weights $\mathbf{w}$ that define the shape of the movement, we propose to apply PI^2. To learn the remaining DMP parameters, i. e. the time duration of the movement τ and the goal of the movement $\mathbf{g}$, and to tune the initial query point $\mathbf{q}$, we use the PoWER algorithm.

B. Augmenting the Example Database

The search process of Section III-A enables the robot to find a new movement $\mathbf{M}$ that solves the task at query point $\mathbf{q}$. We can now expand the example database used to compute generalization function (1) with a pair $(\mathbf{M}, \mathbf{q})$. However, not every movement is suitable to be added to the database. Statistical learning can only be successful if the movements in the database smoothly transition between each other. Since generalization function (1) is smooth, it always generates smooth movement patterns. Unfortunately, in general it is not guaranteed that the optimal movement lies on the manifold defined by the current estimate of the generalization function. Hence we need to allow the robot to search also outside of this manifold. This search can be accomplished by reinforcement learning, but this could result in discontinuities in the movement pattern because the solution is not unique and the robot could find a different type of solution than the ones in the database.

To ensure that this does not occur, we utilize the first movement approximation computed by the generalization function (1). This function computes the reference DMP $\tilde{\mathbf{y}}$, which is guaranteed to transition smoothly between the neighboring DMPs (movements) in the database. Let $\mathbf{y}$ denote the current estimate for the desired movement. We can define the immediate cost function r that can be used by PI^2 as follows

$$r(t) = \kappa \|\mathbf{y}(x(t)) - \tilde{\mathbf{y}}(x(t))\|^2 + \dot{\mathbf{y}}^T \mathbf{\Gamma} \dot{\mathbf{y}}, \qquad (6)$$

where x is the common phase of the DMPs and $\mathbf{\Gamma}$ is the regularization matrix. Such immediate reward ensures that new trajectories generated by PI^2 stay close to the generalized trajectories, which results in smooth transitions between the neighboring trajectories.

PI^2 has only one free parameter, i. e. the exploration noise. In general the exploration noise can be set significantly smaller than usually because most of the exploration is expected to occur in the query point space and not in the full parameter space. We will omit the details of reinforcement learning algorithm PI^2 in this paper. See [5], [7] for more details about PI^2.

Let R be the terminal reward function and let r be the immediate cost function for the desired task. Reinforcement learning in the trajectory space parametrized by $\mathbf{q}$ and its neighborhood can then be defined as follows:

1) Estimate the DMP (first approximation) for a new query point $\mathbf{q}$ using statistical generalization (1). Execute the resulting DMP and compute the terminal reward. Use these data to initialize the PoWER algorithm and use the weights $\mathbf{w}$ of the DMP to initialize PI^2 algorithm.

2) The learning process is stopped if the terminal reward R is above a given threshold or the maximum number of iteration has been reached. Otherwise continue with one reinforcement learning epoch.
3) Repeat K times: obtain a new estimate $\mathbf{q}'$ with exploration noise. For each new $\mathbf{q}'$, compute the DMP parameters using statistical generalization (1) and add the exploration noise. Execute the resulting DMP and calculate the immediate costs and terminal reward function R.
4) Use PoWER to compute new goal $\mathbf{g}$ and time duration τ on the movement and use PI^2 algorithm to compute new weights $\mathbf{w}$ using the results of previous K steps.
5) Use the output of PoWER and PI^2 algorithms and execute the DMP to get the terminal reward R. Continue with step 2.

If the learning process has stopped due to the satisfactory terminal reward, the newly calculated movement can be added to the database of example movements.

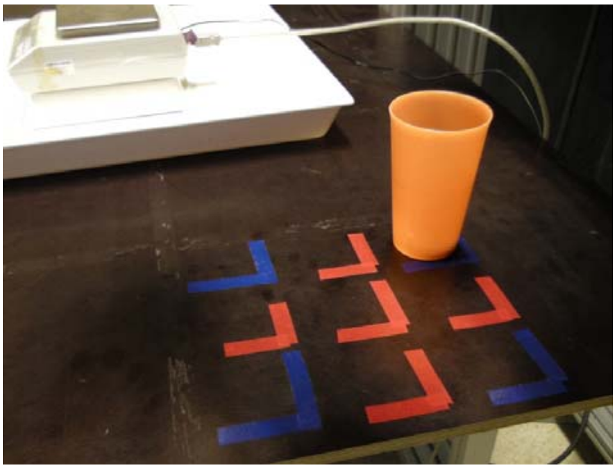

Fig. 2. Experimental setup. Movements for 4 different glass positions (shown in blue) with 2 different quantities of water were initially acquired. In red are the positions on the table that were later added to the database. The scale used in the experiments is in the background.

IV. Experimental results

In our experiments we focused on showing that the proposed approach addresses the following two key issues: 1) the generation of task solution trajectories that are similar to the example trajectories in the existing database, and 2) augmenting the database with autonomously learned trajectories to improve the accuracy of statistical generalization, thereby increasing the overall performance of the task.

The robot's experimental task was to learn how to pour the same quantity of water (0.2 l) into a glass regardless of the amount of water in the bottle (see also Fig. 1). For this purpose we collected 8 initial movements at 4 different glass positions, as shown in Fig. 2, with 2 different quantities of water (0.3 l and 1 l) in the bottle. In this case the query point $\mathbf{q}$ is defined as $[x, y, v]^T$, where x and y denote the position of the glass on the table and v denotes the volume of the water in the bottle. We used the following terminal reward function

$$R = \begin{cases} 5 \cdot (0.2 - |0.2 - m_g| - m_s) & 0 \le R \le 1 \\ 0 & \text{otherwise,} \end{cases} \tag{7}$$

where m_g is the amount of water in the glass and m_s the amount of spilled water. This way we ensure that the robot learns how to pour without spilling. We used a scale to measure the final amount of water in the glass and the force-torque sensor in the wrist to estimate the amount of all water poured from the bottle. The difference between the two quantities provides the amount of spilled water.

The eight task demonstrations were used to provide initial data for statistical generalization. The demonstrated trajectories in joint space were encoded as DMPs with 20 radial basis functions for each joint trajectory. It turned out that 20 is the optimal number of radial basis functions depending on the complexity and length of the pouring movements. Using the learning process described in Section III, 40 new movements together with the associated query points were added to the database. On the average, the reinforcement learning process of Section III needed about 10 executions of the pouring behavior to find a new movement with satisfactory reward. This relatively low number of trials was due to the fact that algorithm can exploit the previously estimated structure of the solution space. Lets analyze the shape of the newly learned movements. Fig. 3 shows the trajectories at 9 different positions on the table for the degree of freedom (5th joint), at which the largest movement was performed during the execution of the pouring behavior. The trajectories exhibit similar shape and transition smoothly from one to another. Fig. 4 show the original and the autonomously learned trajectories in 3-D Cartesian space after being transformed via forward kinematics. Again we can see that the trajectories found via reinforcement learning are qualitatively similar in shape to the demonstrated trajectories. Note that the new trajectories are smoother than the demonstrated trajectories, which is due to the regularization term in reward function (6). Thus we can

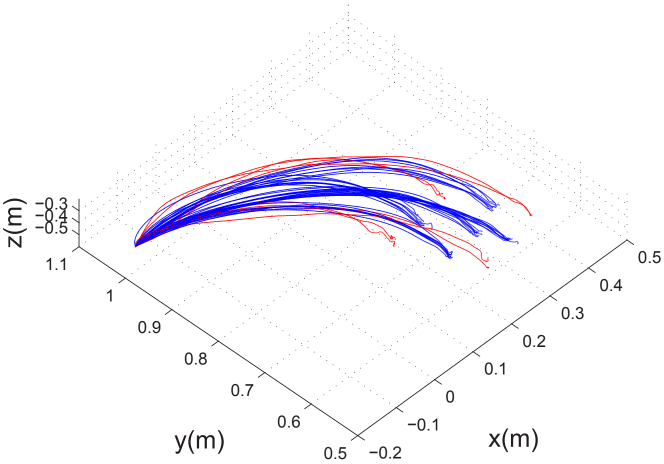

Fig. 4. Similarity of the newly acquired movements. In red are the original example trajectories obtained by kinesthetic guiding and in blue the newly acquired trajectories generated by the proposed reinforcement leaning process. All trajectories were originally determined in joint space and were mapped onto 3-D Cartesian space via forward kinematics.

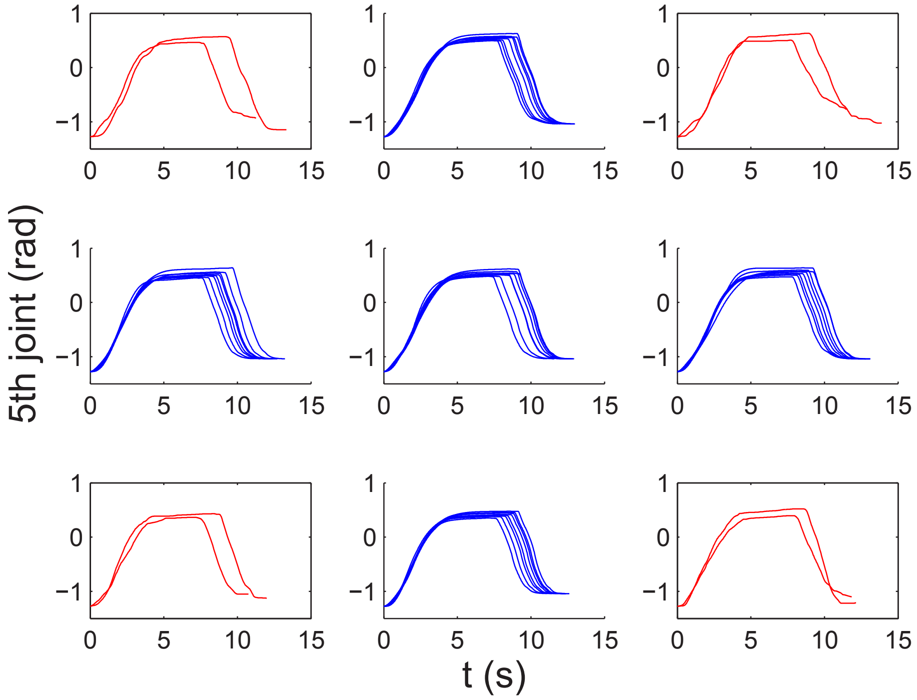

Fig. 3. Similarity of the newly acquired movements. In red are the original example trajectories obtained by kinesthetic guiding and in blue the newly acquired trajectories generated by the proposed reinforcement leaning process. For easier representation only trajectories of 5th joint, that is changing the most, are shown in the graphs.

expect that the augmented movement database will be suitable for statistical generalization.

To evaluate the performance of generalization, we tested the accuracy of generalization at in-between query points (shown in Fig. 5), that is at query points that do not exactly coincide with the training query points. Fig. 6 illustrates the generation of new trajectories by statistical generalization. As expected, the generalized trajectories are smooth and similar to the nearest trajectories in the database. Fig. 5 shows that a considerable increase in performance can be achieved this way. When new test movements were generated by statistical generalization, which used only 8 initial movements as training data, the average error of the glass filling process was 0.08 l. This was reduced to 0.02 l for cases in which statistical generalization used also the newly acquired movements, that is altogether 8 + 40 example movements. The amount of spilled water was also reduced. With the initial 8 movements, statistical generalization produced movements that caused 0.03 l of water to be spilled on the average. With the additional 40 example movements, there was no spilling with generalized DMPs. We can thus claim that the proposed approach is successful at finding new movements for the database and can autonomously increase the performance of generalization.

V. Conclusion

The main result of this paper is a new approach to autonomously augment the database of example movements, which was initially obtained in a supervised manner, for example by human demonstration. The newly acquired data can be used to improve the accuracy of generalization and consequently the performance of task execution. We demonstrated experimentally that using the proposed approach, the robot can improve its performance without additional help of the teacher. We believe that our approach addresses one of the fundamental problems of imitation, that is how to transition from directly mimicking the teacher's movements to practicing, where the initial movements are modified and new data are added to the existing knowledge base.

The integration of autonomous exploration with statistical generalization also enabled us to define a new, structured reinforcement learning algorithm, which can find new example movements in the neighborhood of the estimated trajectory manifold much quicker than standard reinforcement learning algorithms. Compared to the method proposed in [6], which also combined learning in low-dimensional spaces using value function approximation with learning in high-dimensional spaces using PI^2, the approach proposed in this paper uses the results of statistical generalization in addition, thereby further increasing the speed of convergence. The key to this accelerated convergence of the learning process lies in the fact that thanks to statistical generalization function (1), a significant part of the search process could be moved from the high-dimensional trajectory space spanned by DMPs to the low-dimensional space determined by query points, which are defined as task-relevant parameters that characterize the task. Another important issue is that statistical generalization provides a reference movement, which can be used to define immediate rewards in terms of the distance between trajectories determined by reinforcement learning and the reference trajectory. This is needed to ensure that the newly found trajectories are similar to the trajectories in the database.

The developed system is of course not limited to the learn-

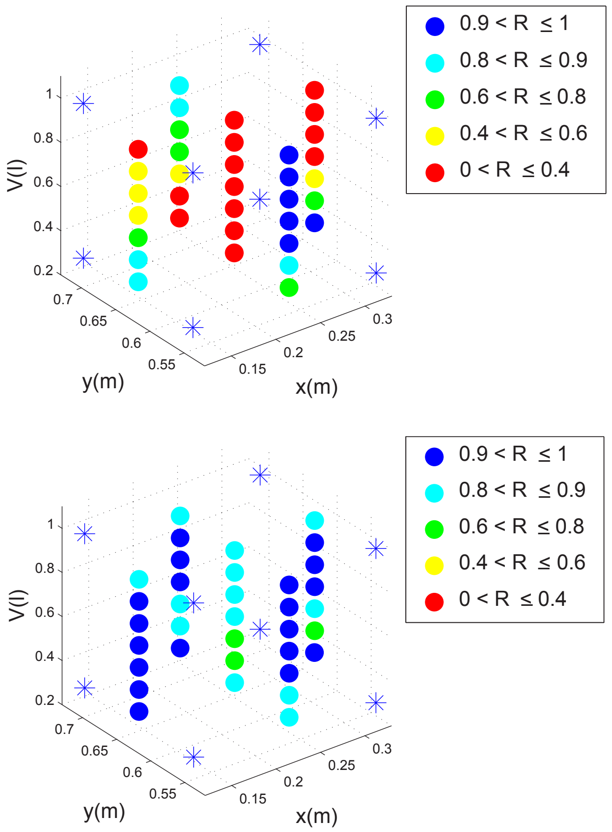

Fig. 5. Results showing the improved performance with the increasing size of the database. Points in the graphs are the test points situated in-between query points. R denotes terminal reward (7), which is normalized to values between zero and one. Demonstrated examples are marked as blue stars and are situated in the corners of the search space. First graph shows terminal rewards at test points where only 8 demonstrated examples were in the trajectory database. Second graph shows terminal rewards at the same test points after additional 40 movements were refined and stored in the trajectory database.

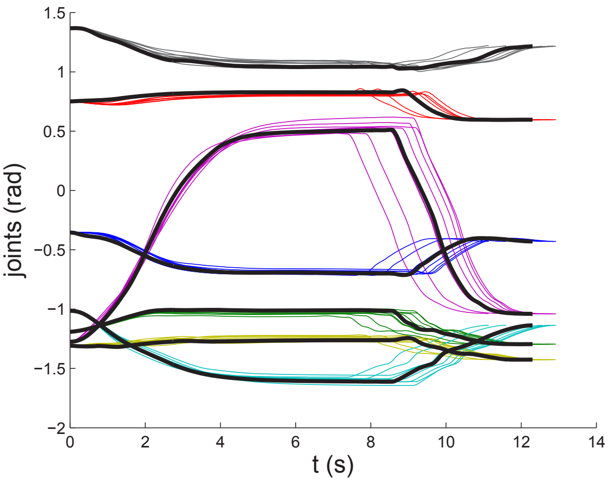

Fig. 6. Calculation of new joint trajectories using statistical generalization. In black are the generalized trajectories and in color the closest neighboring trajectories from the database.

ing of the pouring behaviour. There are only two components that are action-specific: the definition of the query point space and the definition of terminal reward functions R. Note that the immediate reward function r of Eq. (6) is not dependent on the actual behaviour to be learned. Due to space limitations, the analysis in this paper focused on pouring. However, in our previous papers [9], [11], [24] we showed that the algorithms which form the basis of the proposed system can be applied to learn many different behaviors including reaching, throwing, drumming, and kendama.

ACKNOWLEDGEMENTS

Research leading to these results was supported by the European Community's Seventh Framework Programme FP7/2007-2013 (Specific Programme Cooperation, Theme 3, Information and Communication Technologies) grant agreement no. 270273, Xperience.

REFERENCES

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

[2] H. Miyamoto, S. Schaal, F. Gandolfo, H. Gomi, Y. Koike, R. Osu, E. Nakano, Y. Wada, and M. Kawato, "A kendama learning robot based on bi-directional theory," *Neural Networks*, vol. 9, no. 8, pp. 1281–1302, 1996.

[3] J. Morimoto and K. Doya, "Acquisition of stand-up behavior by a real robot using hierarchical reinforcement learning," *Robotics and Autonomous Systems*, vol. 36, pp. 37–53, 2001.

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

[5] E. A. Theodorou, J. Buchli, and S. Schaal, "A generalized path integral control approach to reinforcement learning," *Journal of Machine Learning Research*, vol. 11, pp. 3137–3181, 2010.

[6] M. Tamosiunaite, B. Nemec, A. Ude, and F. Wörgötter, "Learning to pour with a robot arm combining goal and shape learning for dynamic movement primitives," *Robotics and Autonomous Systems*, vol. 59, no. 11, pp. 910–922, 2011.

[7] 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.

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

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

[10] T. Matsubara, S.-H. Hyon, and J. Morimoto, "Learning parametric dynamic movement primitives from multiple demonstrations," *Neural Networks*, vol. 24, no. 5, pp. 493–500, 2011.

[11] D. Forte, A. Gams, J. Morimoto, and A. Ude, "On-line motion synthesis and adaptation using a trajectory database," *Robotics and Autonomous Systems*, vol. 60, pp. 1327–1339, 2012.

[12] K. Mülling, J. Kober, O. Kroemer, and J. Peters, "Learning to select and generalize striking movements in robot table tennis," *International Journal of Robotics Research*, vol. 32, no. 3, pp. 263–279, 2013.

[13] S. Calinon, F. D'halluin, E. L. Sauser, D. G. Caldwell, and A. G. Billard, "Learning and reproduction of gestures by imitation: An approach based on Hidden Markov Model and Gaussian Mixture Regression," *IEEE Robotics and Automation Magazine*, vol. 17, no. 2, pp. 44–54, 2010.

[14] S. M. Khansari-Zadeh and A. Billard, "Learning stable nonlinear dynamical systems with Gaussian Mixture Models," *IEEE Transactions on Robotics*, vol. 27, no. 5, pp. 943–957, 2011.

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

[16] A. J. Ijspeert, J. Nakanishi, and S. Schaal, “Dynamical movement primitives: Learning attractor models for motor behaviors,” *Neural Computation*, vol. 25, no. 2, pp. 328–373, 2013.

[17] C. E. Rasmussen and C. Williams, *Gaussian Processes for Machine Learning*. Cambridge, MA: MIT Press, 2006.

[18] C. G. Atkeson, A. W. Moore, and S. Schaal, “Locally weighted learning for control,” *AI Review*, vol. 11, pp. 75–113, 1997.

[19] J. Kober, “Reinforcement learning for motor primitives,” in *Masters thesis*, University of Stuttgart, Germany, August 2008.

[20] J. Peters and S. Schaal, “Policy gradient methods for robotics,” in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS)*, 2006, pp. 2219–2225.

[21] F. Guenter, M. Hersch, S. Calinon, and A. Billard, “Reinforcement learning for imitating constrained reaching movements,” *Advanced Robotics*, vol. 21, pp. 1521–1544, 2007.

[22] H. J. Kappen, W. Wiegerinck, and B. van den Broek, “A path integral approach to agent planning,” in *AAMAS*, 2007.

[23] B. van den Broek, W. Wiegerinck, and B. Kappen, “Graphical model inference in optimal control of stochastic multi-agent systems,” in *Journal of Artificial Intelligence Research*, vol. 32, 2008, pp. 95–122.

[24] B. Nemec, D. Forte, R. Vuga, M. Tamošiunaite, F. Wörgötter, and A. Ude, “Applying statistical generalization to determine search direction for reinforcement learning of movement primitives,” in *2012 12th IEEE-RAS Int. Conf. Humanoid Robots*, Osaka, Japan, 2012, pp. 65–70.