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

Generalization of Example Movements with Dynamic Systems

Andrej Gams and Aleš Ude

Abstract— **In the past, nonlinear dynamic systems have been proposed as a suitable representation for motor control. It has been shown that it is possible to learn desired complex control policies by a nonlinear transformation of an existing simpler control policy, which is based on a canonical dynamic system. The resulting control policies were termed dynamic movement primitives. The main result of this paper is an approach to learning parametrized sets of dynamic movement primitives based on a library of example movements. Learning was implemented by applying locally weighted regression where the goal of an action is used as a query point into the library of example movements. The proposed approach enables the generation of a wide range of movements that are adapted to the current configuration of the external world without requiring an expert to appropriately modify the underlying differential equations to account for percepetual feedback.**

I. INTRODUCTION

Learning of robotic behaviors that involve manipulation of objects placed at arbitrary locations in the 3D world is difficult because the search space that needs to be explored is potentially huge. It depends both on the number of degrees of freedom of the robot and on the objects directly or indirectly involved in the action. Overcoming the problems arising from high dimensional and continuous perception-action spaces requires reducing the search space, for example by guiding the search process. An effective way is to design a model of the task and learn the specified parameters. However, this relies on externally provided knowledge, e.g. by an engineer, and is therefore unsatisfactory for robots that need to autonomously operate in unstructured environments and have to constantly solve new scenarios. A cognitive robotic system needs to be able to acquire new skills without relying on the presence of the expert.

Autonomous exploration, as one of the ways to acquire new action knowledge, is often studied in the field of developmental robotics [1]. Imitation, as another, also often joined with coaching and practicing, assumes that initial knowledge is available to the robot, often in the form of motor primitives [2]. For example, direct imitation has been successfully applied to learn complex movements on humanoid robots such as dancing, which would be difficult to program manually [3], [4], [5]. Periodic tasks can be performed with imitation by applying appropriate mechanisms that synchronize the motion to a perceptual input [6], [7]. Direct imitation is, however, not useful in object manipulation problems because the observed movements need to be adapted to the current state of the 3-D world. It is highly unlikely that an appropriate movement would be observed in advance and included in the library of observed movements for any given situation. A methodology to generalize the observed movements to new situations was proposed by Miyamoto et al. [8] who developed a new representation for the desired trajectory, which they referred to as via-points.

The work of Miyamoto et al. [8] shows the importance of a proper representation for the control policy. Other explicit, time-dependent representations include splines, which were utilized in [9] to generate new actions from a library of example movements. Hidden Markov models (HMMs) are another popular methodology to encode and generalize the observed trajectories [10], [11], [12]. While techniques that enable the reproduction of generalized movements from multiple demonstrations have been proposed within HMM formalism, generalization across movements to attain an external goal of the task is not central to these works. HMMs, however, can be used effectively for motion and situation recognition [12] and to determine which control variables should be imitated and how [11].

A fundamentally different approach to movement representation based on nonlinear dynamic systems as policy primitives was proposed in [13], [14], [15], [16]. The resulting primitive movements were termed as *dynamic movement primitives* (DMPs). DMPs are based on systems of second-order differential equations, which encode the properties of the desired motion. Ijspeert et al. [13], [14] proposed equations for rhythmic and discrete movements and demonstrated that they can be used to learn tasks such as tennis strokes and drumming. Rhythmic pattern learning was extended to patterns of undetermined frequency in [7]. One of the most important advantages of DMPs is that they remove the direct dependency of the control policy on time. As noted in [16], explicit timing is cumbersome, as it requires maintaining a clocking signal, e. g., a time counter that increments at very small time steps. By removing explicit time dependency, DMPs can easily account for unforeseen perturbations that occur during movement execution without extensive bookkeeping. Another aspect of the approach is the ability to modulate and generalize the learned motion in shape, length and final goal by changing only a small number of parameters [16]. This, however, changes the properties and specific features of the learned motion. Approaches to reduce the change of the modulated motion are described in [17], [18]. A type of generalization between motor primitives is described in [19].

The main purpose of this paper is to propose and ex-

This work was supported in part by the EU Cognitive Systems project PACO-PLUS (FP6-027657)

A. Gams is with Jožef Stefan Institute, Jamova cesta 39, 1000 Ljubljana, Slovenia, andrej.gams@ijs.si

A. Ude is with Jožef Stefan Institute, Jamova cesta 39, 1000 Ljubljana, Slovenia, and ATR Computational Neuroscience Laboratories, 2-2-2 Hikaridai, Soraku-gun, Kyoto 619-0288, Japan ales.ude@ijs.si

perimentally evaluate a method for generalizing example movements to new situations using the dynamic movement primitives as basic representation. While DMPs can be adapted in several ways, generic adaptations cannot account for specific features of the task. The approach proposed in this paper enables the generation of DMPs using a library of example movements together with associated goals of the task and/or other relevant features, which are utilized as query points for generalization.

II. Motion trajectories as second order dynamic systems

Dynamic movement primitives have been introduced in. [13], [14] as means for trajectory generation and modulation. To encode the trajectory of a single variable y, which can either be one of the internal joint angles or one of the external task space coordinates, the following system of linear differential equations with constant coefficients has been proposed as a basis for approximation [16]

$$\tau\dot{z} = \alpha_z(\beta_z(g-y)-z), \quad (1)$$
$$\tau\dot{y} = z. \quad (2)$$

Here y is the trajectory and α_z and β_z are positive constants. τ defines the time-length of the DMP. The system is critically damped for $\alpha_z = 4\beta_z$. Differential equations (1)–(2) ensure that y converges to the goal g and can therefore be used to realize discrete reaching movements. Schaal et al. [16] proposed to add a linear combination of radial basis functions to (1)

$$f(x) = \frac{\sum_{i=1}^{N} w_i\Psi_i(x)}{\sum_{i=1}^{N}\Psi_i(x)}x, \quad (3)$$
$$\Psi_i(x) = \exp\left(-h_i\,(x-c_i)^2\right). \quad (4)$$

Variables Ψ_i and w_i present the basis functions and the weights associated to them, respectively. The parameter $h_i > 0$ represents the width of the basis functions, N their number, and c_i, $i = 1...N$ is equally spaced on x. A new variable x is used in (3) and 4) instead of time to avoid direct dependency of f on time; the dynamics is governed by

$$\tau\dot{x} = -\alpha_x x, \quad (5)$$

with initial value $x_0 = 1$. A solution to (5) is given by $\exp(-\alpha_x t/\tau)$, thus x tends to 0 as time increases. This results in the following differential equation system

$$\tau\dot{z} = \alpha_z(\beta_z(g-y)-z)+f(x) \quad (6)$$
$$\tau\dot{y} = z, \quad (7)$$

which can be used to approximate discrete movements of various shapes. The role of x is also to localize the radial basis functions along the trajectory that needs to be approximated. The trajectory y as specified by (5)–(7) defines what is called a *dynamic movement primitive* (DMP).

III. Motion generalization with dynamic movement primitives

There exists a number of modifications to this equation system in the literature; e. g. Pastor et al. [17], [18] proposed to add terms that enable obstacle avoidance and sequencing of DMPs. These approaches demonstrate why DMPs are useful, but they also require a qualified person to modify the basic equation system in order to adapt the movements to new situations. The main contribution of this paper is an approach that enables modifying the learned trajectories directly from the data, while still encoding the desired movements with dynamic systems.

Dynamic movement primitives are most commonly generated from demonstration trajectories, which are usually acquired by guiding a robotic arm through a sequence of poses or by observing human motion and transforming it to robot motion , see for example [13]. Let $\{y_d(t_j),\dot{y}_d(t_j),\ddot{y}_d(t_j)\}$, $t_j = j\Delta t, j = 0,\ldots,T$, denote a set of positions, velocities and accelerations of the desired trajectory at times t_j. Replacing z in (6) by y results in the following set of equations that need to be solved to calculate a DMP that best fits the data

$$\begin{aligned} \mathrm{F}(t_j) &= \tau^2\ddot{y}_d(t_j)+\alpha_z\tau\dot{y}_d(t_j)-\alpha_z\beta_z(g-y_d(t_j)) \\ &= \frac{\sum_{i=1}^{N} w_i\Psi_i(x_j)}{\sum_{i=1}^{N}\Psi_i(x_j)}x_j, \end{aligned} \quad (8)$$

where $x_j = x(t_j) = \exp(-\alpha_x t_j/\tau)$. Note that as we consider one-shot learning, we have to assume smooth and derivable demonstration trajectories without perturbations that are not part of the original movement. This allows using the analytical solution of (5). In the execution phase, on the other hand, we can still use a modified version of (5) proposed in [13], which enables the trajectory generation system to deal with unforeseen perturbations. In matrix form we have

$$\mathbf{X}\mathbf{w} = \mathbf{f}, \quad (9)$$

where

$$\mathbf{X} = \begin{bmatrix} \frac{\Psi_1(x_1)}{\sum_{i=1}^{N}\Psi_i(x_1)}x_1 & \cdots & \frac{\Psi_N(x_1)}{\sum_{i=1}^{N}\Psi_i(x_1)}x_1 \\ \cdots & \cdots & \cdots \\ \frac{\Psi_1(x_T)}{\sum_{i=1}^{N}\Psi_i(x_T)}x_T & \cdots & \frac{\Psi_N(x_T)}{\sum_{i=1}^{N}\Psi_i(x_T)}x_T \end{bmatrix},$$

$$\mathbf{f} = \begin{bmatrix} \mathrm{F}(t_1) \\ \cdots \\ \mathrm{F}(t_T) \end{bmatrix}, \quad \mathbf{w} = \begin{bmatrix} w_1 \\ \cdots \\ w_N \end{bmatrix}.$$

In the equations above, just as in (1) and (2), α_x, α_z and β_z are constant, g is the desired final position or goal, and τ is set to the time duration of the example movement. What needs to be estimated are the parameters w_i, c_i and h_i. Schaal et al. [16] applied locally weighted projected regression to estimate all these parameters, while other approaches just fix the number of radial basis functions (N) and their extent (h_i) and estimate parameters w_i using regression methods [13]. Locally weighted recursive least squares have proved to be especially useful for incremental learning of rhythmic movements [14], [7].

A. Locally weighted regression

In this paper we show how to generalize DMPs to situations that were not recorded in the example database. We call the process *Locally weighted regression*. The term denotes a common regression technique with which we estimate the parameters of motion from a library of motions.

Lets assume that we have a number of example movements $\{y_d^k(t_j), \dot{y}_d^k(t_j), \ddot{y}_d^k(t_j)\}, t_j = j\Delta t, j = 0,\ldots,T_k,\ k = 1,\ldots,M$, with associated query points $\mathbf{q}_k$ and time constants τ_k. In the example of reaching movements, the query points can be just the final reaching goal $\mathbf{g} = (g_1,\ldots,g_n)$, where n is the dimension of the parameter space. For other movements, query points can differ from $\mathbf{g}$. Even in the case of reaching movements, the query points can be given in Cartesian space, while the DMPs might be encoded in the joint space. The issue is how to generate a DMP representing a new movement for every query $\mathbf{q}$, which in general will not be one of the examples $\mathbf{q}_k$. A rather trivial solution would be to look for a $\mathbf{q}_k$ closest to $\mathbf{q}$, but locally weighted regression enables us to compute a better solution by minimizing criterion

$$\sum_{k=1}^{M} \|\mathbf{X}_k\mathbf{w} - \mathbf{f}_k\|^2 \,\mathrm{K}(\mathrm{d}(\mathbf{q},\mathbf{q}_k)), \tag{10}$$

where K is the kernel function for locally weighted regression and d is the metrics in the space of query points $\mathbf{q}$. Note that even if g of (6) is taken to be the query point – as could be done in the case of reaching movements – it is still worthwile to calculate new DMPs by minimizing criterion (10). This way we can ensure that the new movement has similar properties and specific features to the example movements. Although every DMP eventually converges to g, the course of the trajectory becomes significantly different if the new movement is generated by a DMP whose other parameters were calculated using an example with a significantly different g (see Section IV-A). Locally weighted regression has been thoroughly studied in [20]. It is a form of lazy learning where the computational cost of training is minimal; it simply consists of storing examples in the database.

Unlike α_x, α_z, β_z, N, c_i and h_i, which are kept constant across the example trajectories[1], time constant τ and the goal position g change from example to example. We propose to calculate these parameters from the data as a function of the query point $\mathbf{q}$. Writing

$$\tau = \mathrm{f}_\tau(\mathbf{q}) = \sum_{i=1}^{S_\tau} a_i' \mathrm{B}_i'(\mathbf{q}), \tag{11}$$

$$\mathbf{g} = \mathbf{f}_g(\mathbf{q}) = \sum_{i=1}^{S_g} \mathbf{a}_i'' \mathrm{B}_i''(\mathbf{q}), \tag{12}$$

where B_i' and B_i'' are a suitable set of basis functions, parameters a_i' and a_i'' are estimated by respectively minimizing

$$\sum_{k=1}^{M} |\mathrm{f}_\tau(\mathbf{q}_k) - \tau_k|^2 \text{ and} \tag{13}$$

$$\sum_{k=1}^{M} \left\|\mathbf{f}_g(\mathbf{q}_k) - \mathbf{g}_k\right\|^2. \tag{14}$$

We selected B-splines as basis functions for the approximation of $\mathbf{g}$ and τ. Note that in the training phase, the original τ_k and $\mathbf{g}_k$ are used to calculate $\mathrm{F}_k(t_j)$ defined by (8). In the execution phase, new τ and $\mathbf{g}$ are estimated using transformation functions (11) and (12).

The computational complexity of solving the least squares system given by (10) is $\mathcal{O}(N^2\sum_{k=1}^{M} T_k)$ and thus increases linearly with the number of data points and quadratically with the number of radial basis functions (4) used to represent the DMP. The quadratic dependence on the number of basis functions is not a problem because this number is generally much lower than the number of data points. The complexity is further reduced because $\mathbf{X}_k$ are sparse due to the local support of radial basis functions. These facts make computational complexity low enough to allow resolving the least-squares problem (10) using standard methods from sparse matrix algebra and without resorting to the approximation of $\mathbf{w}$ by local models as implemented in [13], [14]. Incremental learning using recursive least squares is also possible.

The proposed approach is appropriate only if example trajectories smoothly transition as a function of query points. If this is not the case, nearby data does not provide information about the movement associated with the query point $\mathbf{q}$. The above process estimates the parameters $\mathbf{w}, \tau, \mathbf{g}$, which means that the function

$$\mathbf{F}(\mathbf{q}) = (\mathbf{w}, \tau, \mathbf{g}) \tag{15}$$

needs to be smooth.

There are many possibilities to select the weighting function K [20]. We chose the tricube kernel

$$\mathrm{K}(d) = \begin{cases} (1-|d|^3)^3 & \text{if} |d| < 1 \\ 0 & \text{otherwise} \end{cases} \tag{16}$$

This kernel has finite extent and a continuous first and second derivative. Combined with distance d in the space of query points, these two functions determine how much influence each of the example movements $\{y_d^k(t_j), \dot{y}_d^k(t_j), \ddot{y}_d^k(t_j)\}$, $t_j = j\Delta t, j = 0,\ldots,T_k$, has. It is easy to see that the influence of each example movement diminishes with the distance of the query point $\mathbf{q}$ from the data point $\mathbf{q}_k$. In our experiments the query points were given in Euclidean space and we used weighted Euclidean distance to define d

$$\mathrm{d}(\mathbf{q},\mathbf{q}_k) = \|\mathrm{D}(\mathbf{q}-\mathbf{q}_k)\|,\ D = \mathrm{diag}(a_i),\ a_i > 0. \tag{17}$$

Other metrics could be applied if query points are given in different spaces such as for example the special rotation group.

[1] N, c_i and h_i, $i = 1,\ldots,N$, are estimated in a preprocessing step so that every DMP approximates the associated motion trajectory at least up to the predefined accuracy. It is possible to use locally weighted projected regression for this purpose.

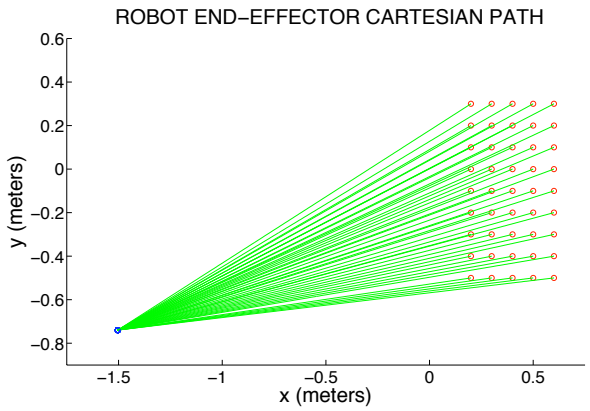

Fig. 1. 45 example minimum jerk trajectories of robot end-effector path in Cartesian space. Red circles depict the final reaching positions that were used as query points for locally weighted regression. The sum of limb lengths was 1.31 meters.

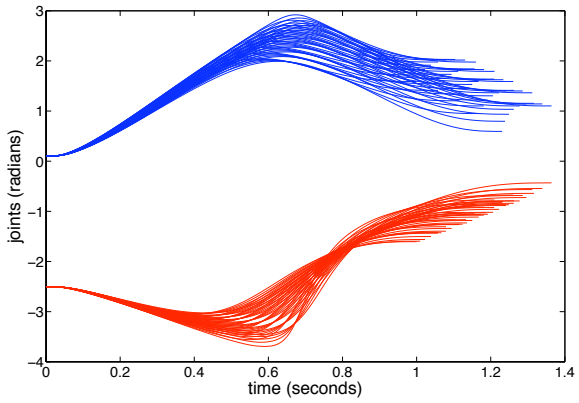

Fig. 2. 45 example joint space trajectories associated with the minimum jerk reaching trajectories in Fig. 1.

IV. Experiments

We examined how well we can approximate Cartesian minimum jerk trajectories of a robot end-effector, which resemble human reaching trajectories [21], by generating DMPs based on the library of example movements given in the robot joint space and using locally weighted regression.

A. Simulated study

In the simulated study we used a planar 2R robot, for which we we first generated Cartesian minimum jerk trajectories that correspond to straight lines. We converted them into joint space trajectories using standard inverse kinematics (see Figs. 1 and 2). The final end-effector positions on the trajectories were used as query points $\mathbf{q}_k$, $k = 1,\ldots,M$, whereas the initial position was the same for all trajectories. We distributed the query points uniformly with spacing of 0.1 meters in a rectangular area with corners at $(0.2,-0.5)$ and $(0.6,0.3)$ meters. Joint velocities and accelerations were computed analytically.

We used B-splines to estimate the mapping (12), which is in this case the mapping from the last end-effector position

TABLE I

Errors in reaching movements (in centimeters and degrees, respectively) synthesized by locally weighted regression.

	Joint space (across trajectory)		Cartesian space (final pos. err.)		Grid size (cm)
Training	Full	Reduced	Full	Reduced	
Average err.	0.24	0.19	0.12	0.09	10×10
Max. error	0.97	0.46	0.47	0.30	10×10

TABLE II

Errors in reaching movements (in centimeters and degrees, respectively) generated by a single DMP.

	Joint space (across trajectory)		Cartesian space (final pos. err)		Grid size (cm)
Training	Full	Reduced	Full	Reduced	
Average err.	8.85	5.62	0.43	0.32	10×10
Max. error	22.47	13.88	0.97	0.77	10×10

to the corresponding joint angles and is thus a local approximation for the inverse kinematics. Similarly we estimated the mapping (11) from query points $\mathbf{q}$ to the time duration of the trajectory. In simulation the time duration was assumed to depend on the distance of the end effector's final position from its initial position on the trajectory.

The errors in Tab. I and II were calculated by integrating (5) – (7) to obtain joint positions $\tilde{\mathbf{y}}(t_j)$, and comparing the result with the ideal minimum jerk trajectory $\mathbf{y}(t_j)$ expressed in the robot joint space. These ideal trajectories were calculated using the same formulas as in the calculation of the training examples. Both average (18) and maximum error (19) on the trajectory were evaluated

$$\frac{1}{N}\sum_{j=0}^{N}\|\tilde{\mathbf{y}}(t_j)-\mathbf{y}(t_j)\|, \tag{18}$$

$$\max_{j=0}^{N}\|\tilde{\mathbf{y}}(t_j)-\mathbf{y}(t_j)\|. \tag{19}$$

Results in Tab. I prove that reaching movements can be approximated by locally weighted regression and DMPs with high precision. Since it can be expected that the errors will be larger on the boundary of the training query points $\mathbf{q}_k$, we estimated the error both within the full rectangular area enclosed by all query points of Fig. 1, and in the reduced area enclosed by query points situated at least one query point away from the boundary points. As expected, the errors are smaller for the internal points. The resulting trajectory accurately reproduced both the spatial course of movement and its dynamics. In this experiment the Cartesian positions were used as query points, which were then mapped to the goal configurations using the estimated spline function (12). This function partially approximates the inverse kinematics as relevant for the task. Thus with the proposed approach we can generalize reaching movements in Cartesian space without knowing the inverse kinematics of the robot.

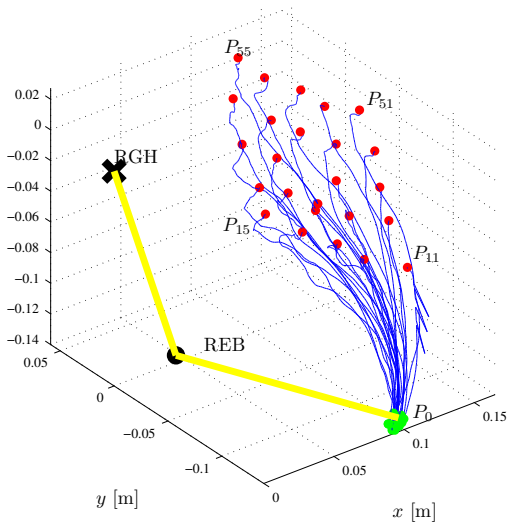

Fig. 3. Grid and trajectories for the HOAP-3 robot reaching example. The goals are marked with red dots, the starting point with a green dots and the trajectories with blue lines. The robot's arm with glenohumeral (RGH) and elbow (REB) joint is in yellow.

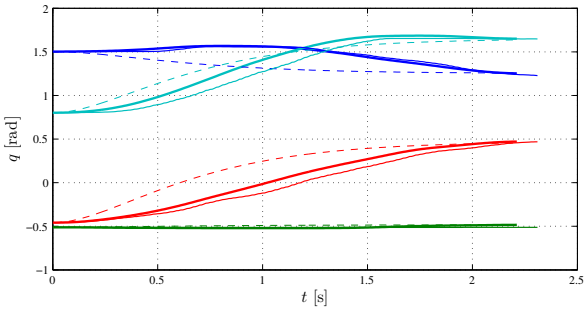

Fig. 4. Real-world experiment results. For a random query point the learned joint trajectories (bold) resemble the trajectories of the nearest point (solid) in the grid. Results using a DMP with a changed goal (dotted) reach the final goal yet do not preserve the specifics of the motion.

The accuracy is sufficient to realize high-precision reaching movements for grasping.

Tab. (II) shows that representation with only one DMP is too rough for precise movement reproduction. While the final position could be reached fairly accurately within the given time due to the properties of discrete movement primitives, the trajectory reproduction accuracy (18) is worse by an order of magnitude compared to the precision of the approach based on locally weighted regression. In both cases the trajectory did not fully reach the final destination because time was not allowed to flow beyond τ estimated by (11). If we allowed the time to flow and continued to integrate the underlying differential equations, the motion would continue and the desired position would eventually be reached, yet with a certain delay.

B. Real world experiments

We tested the same reaching study on a humanoid HOAP-3 robot. We recorded a set of 25 demonstration reaching movements, that served as our library of example movements. All the movements originated from a predefined starting position (P_0) and ended on a grid roughly in the coronal plane of the robot, $\tilde{0}.15$ m in front of the robot. The final reaching goals were on a grid with corners at $P_{11} = (0.167, -0.055, -0.089)$, $P_{15} = (0.153, 0.059, -0.106)$, $P_{51} = (0.127, -0.060, 0.024)$ and $P_{55} = (0.117, 0.044, 0.011)$ meters from the right glenohumeral joint (RGH). The grid and the task space trajectories of the movements are presented in Fig. 3.

Just as in the simulated experiments, we estimated the mapping from the last end-effector to the corresponding joint angle, in this case 4 joint coordinates were used. Again we estimated the mapping (11) from query points $\mathbf{q}$ to the time duration of the trajectory. The time duration was also assumed to depend on the distance of the end effector's final position from its initial position on the trajectory, just as in simulation. Fig. 4 shows the real world experiment results that show that the generalized movement preserves the specific movement features better than a single DMP.

V. Summary and Conclusion

In the paper we propose a movement generalization approach realized using locally weighted regression that defers learning to the execution time. Contrary to many other approaches to modifying DMPs with respect to the perceptual input that require an expert in order to modify the underlying system of differential equations, our approach uses perceptual input as the query into the data and generates the movements directly. The computing time necessary to generate a DMP given a particular query point depends linearly on the number of query points and the number of samples associated with each example movement. Since the weighting criterion has finite support, the number of example trajectories involved in the generation of each DMP remains limited. Therefore the increase in the computing time required for locally weighted regression compared to the computing time needed for standard one-shot learning of discrete DMPs is also limited. An efficient implementation is, however, very important if DMPs are to be generated online using current perceptual input.

The experimental results show that by using DMPs and locally weighted regression, we can accurately approximate a space of smooth movements, e. g. minimum jerk reaching movements. While locally weighted regression can be applied to other representations such as B-splines [9], it has been shown that in the execution phase, dynamic systems have many advantages. This is due to the robustness to perturbations [14], [16]. The proposed approach enables execution of nearly optimal trajectories when there are no disturbances, while keeping the ability to easily adapt the trajectories if necessary. On the other hand, if precision is paramount and any perturbations would lead to failure, for example pure feedforward movements that constitute throwing, splines might be preferable because they do not suffer from problems associated with the integration.

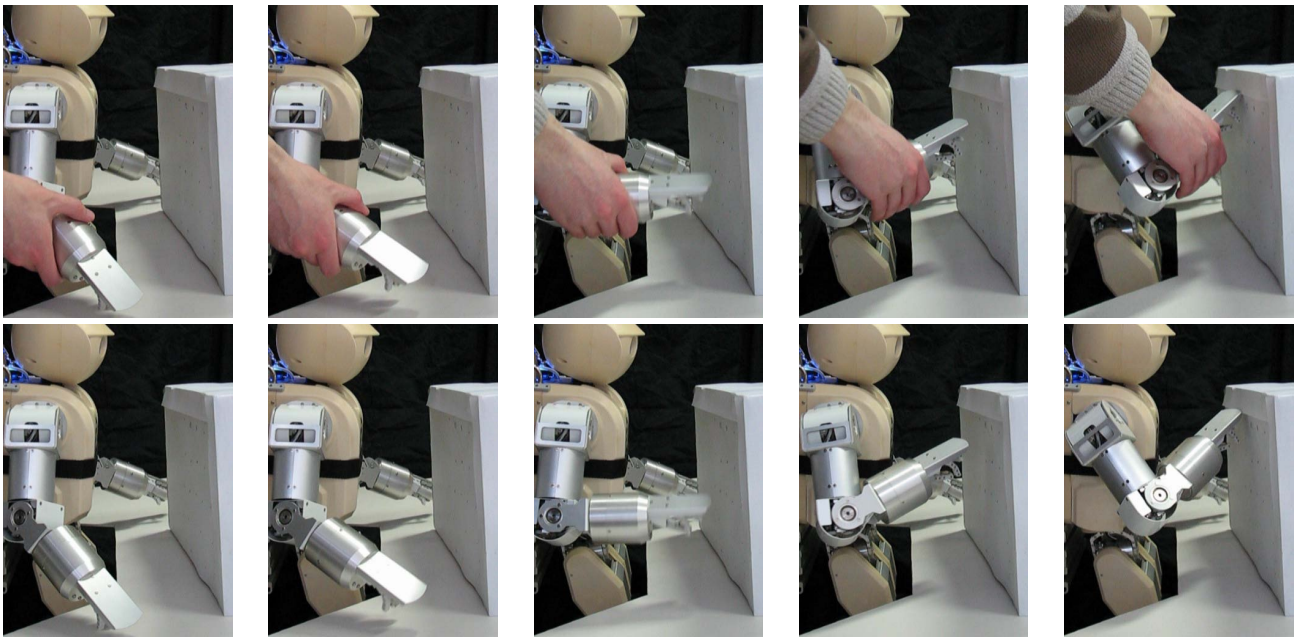

Fig. 5. A sequence of stills showing the teaching (top) and the execution of the generalized movement to a goal within the grid of the example movements.

The generalization of movements (Section III-A) makes sense only for problems with example movements that transition smoothly as a function of query points, which is not always the case. If reaching movements need to avoid an obstacle before arriving to the final configuration, and there are two sets, each avoiding the obstacle from a different side, then example movements that avoid the obstacle from different sides should not be blended together. In such cases the approach described in this paper can still be used, but it should be supplemented by a suitable clustering procedure which must be guided by higher level cognitive processes.

REFERENCES

[1] Max Lungarella, Giorgio Metta, Rolf Pfeifer, and Giulio Sandini. Developmental robotics: a survey. *Connection Science*, 15(4):151–190, 2003.

[2] Stefan Schaal. Is imitation learning the route to humanoid robots? *Trends in Cognitive Sciences*, 3(6):233–242, 1999.

[3] M. Riley, A. Ude, and C. G. Atkeson. Methods for motion generation and interaction with a humanoid robot: Case studies of dancing and catching. In *Proc. 2000 Workshop on Interactive Robotics and Entertainment*, pages 35–42, Pittsburgh, Pennsylvania, April/May 2000.

[4] Aleš Ude, Christopher G. Atkeson, and Marcia Riley. Programming full-body movements for humanoid robots by observation. *Robotics and Autonomous Systems*, 47(2-3):93–108, 2004.

[5] Miti Ruchanurucks, Shinichiro Nakaoka, Shunsuke Kudoh, and Katsushi Ikeuchi. Humanoid robot motion generation with sequential physical constraints. In *Proc. IEEE Int. Conf. Robotics and Automation*, pages 2649–2654, Orlando, Florida, 2006.

[6] Andrej Gams, Leon Žlajpah, and Jadran Lenarčič. Imitating Human Acceleration of a Gyroscopic Device. *Robotica*, 25(4):501–509, 2007.

[7] Andrej Gams, Auke J. Ijspeert, Stefan Schaal, and Jadran Lenarčič. On-line learning and modulation of periodic movements with nonlinear dynamical systems. *Autonomous Robots*, 2009, DOI 10.1007/s10514-009-9118-y.

[8] Hiroyuki Miyamoto, Stefan Schaal, Francesca Gandolfo, Hiroaki Gomi, Yasuharu Koike, Reiko Osu, Eri Nakano, Yasuhiro Wada, and Mitsuo Kawato. A kendama learning robot based on bi-directional theory. *Neural Networks*, 9(8):1281–1302, 1996.

[9] Aleš Ude, Marcia Riley, Bojan Nemec, Andrej Kos, Tamim Asfour, and Gordon Cheng. Synthesizing goal-directed actions from a library of example movements. In *Proc. IEEE-RAS Int. Conf. Humanoid Robots*, Pittsburgh, Pennsylvania, December 2007.

[10] Tamim Asfour, Florian Gyarfas, Pedram Azad, and Rüdiger Dilmann. Imitation learning of dual-arm manipulation tasks in humanoid robots. *International Journal of Humanoid Robotics*, 5(2):183–202, 2008.

[11] Aude Billard, Sylvain Calinon, and Florent Guenter. Discriminative and adaptive imitation in uni-manual and bi-manual tasks. *Robotics and Autonomous Systems*, 54:370–384, 2006.

[12] Tetsunari Inamura, Iwaki Toshima, Hiroaki Tanie, and Yoshihiko Nakamura. Embodied symbol emergence based on mimesis theory. *Int. J. Robotics Research*, 23(4-5):363–377, 2004.

[13] Auke Jan Ijspeert, Jun Nakanishi, and Stefan Schaal. Movement imitation with nonlinear dynamical systems in humanoid robots. In *Proc. IEEE Int. Conf. Robotics and Automation*, pages 1398–1403, Washington, DC, 2002.

[14] Auke Jan Ijspeert, Jun Nakanishi, and Stefan Schaal. Learning attractor landscapes for learning motor primitives. In S. Becker, S. Thrun, and K. Obermayer, editors, *Advances in Neural Information Processing Systems 15*, pages 1547–1554. MIT Press, Cambridge, Mass., 2003.

[15] Stefan Schaal, Jan Peters, Jun Nakanishi, and Auke Ijspeert. Learning movement primitives. In P. Dario and R. Chatila, editors, *Robotics Research: The Eleventh International Symposium*, pages 561–572, Berlin, Heidelberg, 2005. Springer.

[16] Stefan Schaal, Peyman Mohajerian, and Auke Ijspeert. Dynamics systems vs. optimal control – a unifying view. *Progress in Brain Research*, 165(6):425–445, 2007.

[17] Peter Pastor, Heiko Hoffmann, Tamim Asfour, and Stefan Schaal. Learning and generalization of motor skills by learning from demonstration. In *Proc. IEEE Int. Conf. Robotics and Automation*, Kobe, Japan, 2009 (to appear).

[18] Dae-Hyung Park, Heiko Hoffmann, Peter Pastor, and Stefan Schaal. Movement reproduction and obstacle avoidance with dynamic movement primitives and potential fields. In *Proc. IEEE-RAS Int. Conf. Humanoid Robots*, Daejeon, Korea, 2008.

[19] Jan Peters, Jens Kober, Katharina Muelling, Duy Nguyen-Tuong, and Oliver Kroemer. Towards motor skill learning for robotics. In *Proc. Intrnational Symposium on Robotics Research (ISRR)*, Lucerne, Switzerland, 2009.

[20] Christopher. G. Atkeson, Andrew W. Moore, and Stefan Schaal. Locally weighted learning. *AI Review*, 11:11–73, 1997.

[21] Tamar Flash and Neville Hogan. The coordination of arm movements: an experimentally confirmed mathematical model. *The Journal of Neuroscience*, 5(7):1688–1703, 1985.