Project no.: **027657**

Project full title: **Perception, Action & Cognition through Learning of Object-Action Complexes**

Project Acronym: **PACO-PLUS**

Deliverable no.: D2.3.1

Title of the deliverable: Scientific publication on goal-directed action synthesis

Contractual Date of Delivery to the CEC:	January 31st, 2009
Actual Date of Delivery to the CEC:	February 6th, 2009
Organisation name of lead contractor for this deliverable:	JSI
Author(s):	Aleš Ude and Tamim Asfour,
Participants(s):	JSI, UniKarl
Work package contributing to the deliverable:	WP1, WP2, WP8
Nature:	R & D
Version:	1.0
Total number of pages:	4
Start date of project:	1st Feb. 2006 **Duration:** 48 months

Project co-funded by the European Commission within the Sixth Framework Programme (2002-2006)		
Dissemination Level		
PU	Public	**X**
PP	Restricted to other programme participants (including the Commission Services)	
RE	Restricted to a group specified by the consortium (including the Commission Services)	
CO	Confidential, only for members of the consortium (including the Commission Services)	

Abstract:

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 results of this deliverable are an approach to movement generalization by learning parameterized sets of dynamic movement primitives based on a library of example movements and a method that enables sequencing of movement primitives and their adaptation to achieve obstacle avoidance. Both approaches associate object-oriented semantic information with the action primitives, which is essential to enable planning of action sequences. The proposed approaches were tested on various example tasks including reaching, throwing, pouring, and continuous sequencing of the available primitives.

The deliverable comprises one accepted conference paper and one journal paper that is prepared for submission.

Keyword list: Nonlinear dynamic systems, dynamic movement primitives, imitation, sensorimotor learning

Table of Contents

Report Summary

Workpackage 2 focuses on sesnorimotor representations that are implementable on (humanoid) robots. This report describes our work on learning approaches that associate a goal of an action with the internal movement representation to generate new movements and sequences of movements. We experimented both with spline representations [1] and dynamic systems that form the basis for dynamic movement primitives (DMPs) [2]. By associating low-level motor control representations with the goal of an action, we provide an embodied action representation for object-action complexes and also lay the groundwork for sequencing and other higher-level cognitive processes.

In this deliverable we study dynamic movement primitives as a basic representation for goal-directed actions. The main advantage of dynamic movement primitives is that they can be easily modified on-line to account for unforeseen perturbations using feedback control. This is much more difficult to realize when non-autonomous representations such as splines and the related hidden Markov models are used.

Generalization of example movements to new goals with dynamic systems

In paper [B] we focus on how to generalize movements to account for new configurations of the external world. The generalized movements are represented by dynamic systems. The proposed approach is based on locally weighted regression that defers learning to the execution time. It generates new movements directly from the data by combining several localized example movements with the current goal of an action, which is used as a query point. In the execution phase, query points should be provided by perception and the associated dynamic movement primitive can be generated on-line. Unlike our approach, other techniques, which were proposed in the literature to modify DMPs to account for the external perceptual input, require the modification of the underlying system of differential equations. This can only be accomplished by an expert.

Experimental results demonstrate that by using DMPs and locally weighted regression, we can approximate various spaces of smooth movements, e.g. minimum jerk reaching movements, with high accuracy. While locally weighted regression can be applied to other representations such as B-splines [1], nonautonomous representations including splines cannot be adapted to unforeseen perturbations as easily as DMPs. The proposed approach thus enables the robot to execute optimal movements when there are no disturbances, while keeping the ability to easily adapt the movement trajectories if necessary. On the other hand, if precision is paramount and any perturbations would lead to failure, such as in the case of purely feedforward movements that constitute throwing, splines proposed in our earlier work [1] might be preferable because they do not suffer from numerical problems associated with the integration of differential equations.

Learning action sequences that allow object manipulation

In paper [A] we extended the framework of dynamic movement primitives to action sequences that allow object manipulation. We suggested several improvements of the original movement primitive framework and added semantic information to movement primitives, such that they can code object-oriented actions. We demonstrated the feasibility of our approach in an imitation learning setting, where a robot learned a water-serving and a pick-and-place task from a human demonstration, and could generalize this task to novel situations.

We exploited the robustness of dynamic movement primitives against perturbations for obstacle avoidance, which was realized by adding a coupling term to the underlying differential equations of motion. The ability to avoid obstacles in Cartesian space was demonstrated in experiments that included continuous sequences of pick and place operations.

Attached Papers

[A] P. Pastor, H. Hoffmann, T. Asfour, and S. Schaal. Learning and generalization of motor skills by learning from demonstration. In Proc. IEEE Int. Conf. on Robotics and Automation, Kobe, Japan, 2009 (to appear).

[B] A. Ude, A. Gams, and T. Asfour. Generalization of Example Movements with Dynamic Systems. To be submitted, 2009.

References

[1] A. Ude, M. Riley, B. Nemec, A. Kos, T. Asfour, and G. Cheng. Synthesizing goal-directed actions from a library of example movements, Proc. IEEE-RAS Int. Conf on Humanoid Robots, Pitssburgh, Pennsylvania, December 2007.

[2] 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.

Generalization of Example Movements with Dynamic Systems

Aleš Ude[a,c], Andrej Gams[a], Tamim Asfour[b]

[a] *Jožef Stefan Institute, Jamova 39, 1000 Ljubljana, Slovenia*
[b] *University of Karlsruhe, Kaiserstr. 12, 76131 Karlsruhe, Germany*
[c] *ATR Computational Neuroscience Laboratories, 2-2-2 Hikaridai, Soraku-gun, Kyoto 619-0288, Japan*

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.

Key words: Sensorimotor learning, movement primitives, motion blending

1. Introduction

Learning of behaviors that can be applied to solve a given task regardless of the current configuration of the external world is a difficult problem because the search space that needs to be explored is potentially huge. The size of the search space depends both on the number of degrees of freedom of the robot and on the objects involved in the action or affecting it indirectly. To overcome problems arising from high dimensional and continuous perception-action spaces, it is necessary to guide the search process, thus

effectively reducing the search space. The most effective way of reducing the search space is to design a physical model of the task and learn the appropriate parameters. However, such an approach relies on the knowledge that is externally provided by an engineer and is therefore unsatisfactory for robots that need to operate in natural environments and have to solve new problems every day. A cognitive robotic system needs to be able to learn new skills without relying on the presence of the experts.

A robot can acquire new action knowledge in various way. Autonomous exploration is often studied in the field of developmental robotics [1]. Imitation, often connected with coaching and practicing, is another approach which assumes that more initial knowledge is available to the robot, often in the form of motor primitives [2]. For example, direct imitation has been applied successfully to learn complex movements on humanoid robots such as dancing, which would be difficult to program manually [3, 4, 5]. Direct imitation is, however, not useful in problems that involve the manipulation of objects because in such tasks the observed movements need to be adapted to the current state of the 3-D world. For any given situation, it is highly unlikely that an appropriate movement would be observed in advance and included in the library of observed movements. A methodology to generalize the observed movements to new situations was proposed by Miyamoto et al. [6] who developed a new representation for the desired trajectory, which they referred to as via-points. By monitoring the performance of the robot, they were able to adapt the via points so that the robot could play a fairly difficult Japanese game kendama and execute tennis serves. Riley et. al. [7] showed how the observed trajectories can be adapted by coaching and demonstrated the proposed approach on learning how to throw a ball into a basket. These methods, however, require an extensive learning procedure to solve the task in each different configuration of the external world and can thus only be viewed as paradigms to acquire action knowledge for one particular configuration of the external world.

The work of Miyamoto et al. [6] shows the importance of a proper representation for the control policy that can be utilized by the robot to learn how to solve the task. Other explicit, time-dependent representations include splines, which were utilized in [8] 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 [9, 10, 11]. 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 [11] and to determine which control variables should be imitated and how [10].

A fundamentally different approach to movement representation based on nonlinear dynamic systems as policy primitives was proposed in [12, 13, 14, 15]. 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. [12, 13] proposed equations for rhythmic and discrete movements and demonstrated that they can be used to learn tasks such as tennis strokes and drumming. One of the most important advantages of DMPs is that they remove the direct dependency of the control policy on time. As noted in [15], explicit timing is cumbersome, as it requires to maintain a clocking signal, e. g., a time counter that increments at very small time steps. By removing explicit time dependency, DMPs can account for unforeseen perturbations that occur during movement execution without extensive bookkeeping, which is necessary if the desired trajectory is time dependent.

The main purpose of this paper is to propose and experimentally 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 the library of example movements together with the associated goals of the task and/or other relevant features, which are utilized as query points for generalization.

2. Motion trajectories as second order dynamic systems

Dynamic movement primitives have been introduced in. [12, 13] as means for trajectory generation and trajectory 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 [15]

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

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

For the sake of completeness we briefly explain why such a system is useful by examining its general solution. The general solution can be written as a sum of the particular and homogeneous solution, $[z, y]^T = [z_p, y_p]^T + [z_h, y_h]^T$. Here $[z_p, y_p]^T$ is any function that solves Eq. (1) – (2), while $[z_h, y_h]^T$ is a general solution to the system of homogeneous linear equations

$$\begin{bmatrix} \dot{z} \\ \dot{y} \end{bmatrix} = \frac{1}{\tau} \begin{bmatrix} \alpha_z(-\beta_z y - z) \\ z \end{bmatrix} = \mathbf{A} \begin{bmatrix} z \\ y \end{bmatrix}, \ \mathbf{A} = \frac{1}{\tau} \begin{bmatrix} -\alpha_z & -\alpha_z\beta_z \\ 1 & 0 \end{bmatrix}. \quad (3)$$

It is trivial to check that constant function $[z_p(t), y_p(t)] = [0, g]^T$ solves the Eq. system (1) – (2). Moreover, it is well known that a general solution to the homogeneous system (3) is given by $[z_h(t), y_h(t)]^T = \exp(\mathbf{A}t)\,\mathbf{c}$. Thus a general solution to (1) – (2) can be written as

$$\begin{bmatrix} z(t) \\ y(t) \end{bmatrix} = \begin{bmatrix} 0 \\ g \end{bmatrix} + \exp(\mathbf{A}t)\,\mathbf{c}, \quad (4)$$

where $\mathbf{c} \in \mathbb{R}^2$ should be calculated from initial conditions. The eigenvalues of $\mathbf{A}$ are given by $\lambda_{1,2} = \left(-\alpha_z \pm \sqrt{\alpha_z^2 - 4\alpha_z\beta_z}\right)/(2\tau)$. Solution (4) converges to $[0, g]^T$ if the real part of $\lambda_{1,2}$ is smaller than 0, which is true for any $\alpha_z, \beta_z, \tau > 0$. The system is critically damped, which means that y converges to g without oscillating and faster than for any other choice of $\mathbf{A}$, if $\mathbf{A}$ has a double negative eigenvalue. This happens for $\alpha_z = 4\beta_z$.

Differential equations (1) – (2) ensure that y converges to g and can therefore be used to realize discrete reaching movements. To increase a rather simple set of trajectories defined by (4) and thus enable the approximation of any smooth reaching movement, Eq. (1) needs to be modified. Schaal et al. [15] proposed to add a linear combination of radial basis functions to (1)[1]

$$f(x) = \frac{\sum_{i=1}^N w_i \Psi_i(x)}{\sum_{i=1}^N \Psi_i(x)} x, \ \Psi_i(x) = \exp\left(-h_i\,(x - c_i)^2\right), \ h_i > 0. \quad (5)$$

[1] f defined in [15] is scaled by $g - y_0$, i. e. $f(x) = \frac{\sum_{i=1}^N w_i \Psi_i(x)}{\sum_{i=1}^N \Psi_i(x)} x(g - y_0), \ y_0 = y(0)$. Thus when the goal configuration g changes, the encoded movement gets scaled. We omit this scaling factor because we are not interested in automatic scaling, which is achieved differently in our approach. If g is kept constant, the scaling factor has no effect because it gets incorporated in w_i.

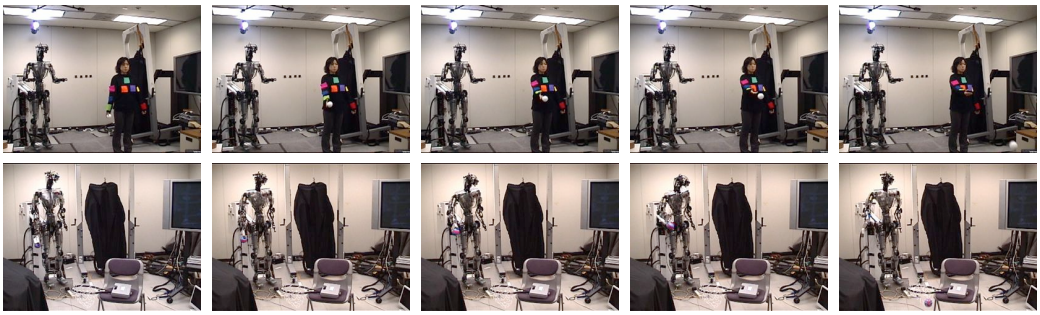

Figure 1: Human demonstration of the ball throw and a successful action execution after coaching

A new variable x is used in (5) instead of time to avoid direct dependency of f on time. Its dynamics is assumed to be

$$\tau \dot{x} = -\alpha_x x, \tag{6}$$

with initial value $x_0 = 1$. A solution to (6) is given by $\exp(-\alpha_x t/\tau)$, thus x tends to 0 as time increases. This results in the following differential equation system

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

$$\tau \dot{y} = z, \tag{8}$$

which can be used to approximate discrete movements of various shapes. Since x tends to zero, the influence of the nonlinear term $f(x)$ decreases with time and system (7) – (8) converges to $[0, g]^T$ just like (1) – (2). The other role of x is to localize the radial basis functions along the trajectory that needs to be approximated. The trajectory of y as specified by differential equations (6) – (8) defines what is called a *dynamic movement primitive* (DMP).

3. Motion generalization with dynamic movement primitives

There exists a number of modifications to this equation system in the literature; e. g. Pastor et al. [16, 17] proposed to add terms that enable obstacle avoidance and sequencing of DMPs. Approaches like this demonstrate why DMPs are useful, but such approaches 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

the modification of the learned trajectories directly from the data, while still encoding the desired movements with dynamic systems.

Dynamic movement primitives are most commonly generated from example 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, possibly adapted by practicing and coaching (see Fig. 1). Let $\{y_d(t_j), \dot{y}_d(t_j), \ddot{y}_d(t_j)\}$, $t_j = j\Delta t, j = 0, \ldots, T$, be a set of positions, velocities and accelerations on the desired trajectory at times t_j. Replacing z in Eq. (7) by y results in the following set of equations that need to be solved to calculate a DMP that best fits the data

$$\mathrm{F}(t_j) = \tau^2 \ddot{y}_d(t_j) + \alpha_z \tau \dot{y}_d(t_j) - \alpha_z \beta_z (g - y_d(t_j)) = \frac{\sum_{i=1}^{N} w_i \Psi_i(x_j)}{\sum_{i=1}^{N} \Psi_i(x_j)} x_j, \quad (9)$$

where $x_j = x(t_j) = \exp(-\alpha_x t_j/\tau)$.[2] In matrix form we have

$$\mathbf{X}\mathbf{w} = \mathbf{f}, \quad (10)$$

where

$$\mathbf{X} = \begin{bmatrix} \frac{\Psi_1(x_1)}{\sum_{i=1}^{N} \Psi_i(x_1)} x_1 & \cdots & \frac{\Psi_N(x_1)}{\sum_{i=1}^{N} \Psi_i(x_1)} x_1 \\ \cdots & \cdots & \cdots \\ \frac{\Psi_1(x_T)}{\sum_{i=1}^{N} \Psi_i(x_T)} x_T & \cdots & \frac{\Psi_N(x_T)}{\sum_{i=1}^{N} \Psi_i(x_T)} x_T \end{bmatrix}, \ \mathbf{f} = \begin{bmatrix} \mathrm{F}(t_1) \\ \cdots \\ \mathrm{F}(t_T) \end{bmatrix}, \ \mathbf{w} = \begin{bmatrix} w_1 \\ \cdots \\ w_N \end{bmatrix}.$$

In the equations above α_x, α_z and β_z are constant, g is the desired final position, 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. [15] applied locally weighted projected regression to estimate all these parameters, while other approaches just fix the number of radial basis functions and their extent and estimate parameters w_i using regression methods [12]. Locally weighted recursive least squares have proved to be especially useful for incremental learning of rhythmic movements [13, 18].

[2]For one-shot learning considered in this paper, we need to assume that example trajectories are good, i. e. they do not contain any perturbations that are not part of the original movement. In this case we can use the analytical solution of (6). However, in the execution phase we can still use a modified version of (6) proposed in [12], which enables the trajectory generation system to deal with unforeseen perturbations.

3.1. Locally weighted regression

In this paper we are interested in how to generalize DMPs to situations that were not recorded in the example database. 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 the associated query points $\mathbf{q}_k$ and time constants τ_k. If we take the example of reaching movements, the query points can be just the final reaching destination $\mathbf{g} = (g_1, \ldots, g_n)$, 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

$$\sum_{k=1}^{M} \|\mathbf{X}_k\mathbf{w} - \mathbf{f}_k\|^2 \,\mathrm{K}(\mathrm{d}(\mathbf{q}, \mathbf{q}_k)), \tag{11}$$

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 Eq. (7) 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 (11). In this way we can ensure that the new movement is similar 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 4.1). Locally weighted regression has been thoroughly studied in [19]. 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[3], time constant τ and the goal position g change from example to example. We propose to calculate these parameters from the data

[3] 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.

as a function of the query point $\mathbf{q}$. Writing

$$\tau = \mathrm{f}_\tau(\mathbf{q}) = \sum_{i=1}^{S_\tau} a'_i \mathrm{B}'_i(\mathbf{q}), \tag{12}$$

$$\mathbf{g} = \mathbf{f}_g(\mathbf{q}) = \sum_{i=1}^{S_g} \mathbf{a}''_i \mathrm{B}''_i(\mathbf{q}), \tag{13}$$

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

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

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 Eq. (9). In the execution phase, new τ and $\mathbf{g}$ are estimated using transformation functions (12) and (13).

The computational complexity of solving the least squares system (11) 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 (5) 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 us to resolve the least-squares problem (11) using standard methods from sparse matrix algebra and without resorting to the approximation of $\mathbf{w}$ by local models as implemented in [12, 13]. 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. Otherwise nearby data does not provide information about the movment 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. In the next section we give a few examples with a smooth movement transition.

There are many possibilities to selet the weighting function K [19]. 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.

4. Experiments

We conducted two types of experiments to demonstrate the usefulness of the approach: the learning of reaching movements and ball throwing.

4.1. Reaching

In a simulation study we examined how well we can approximate Cartesian minimum jerk trajectories of a robot end-effector by generating DMPs based on the library of example movements given in the robot joint space and using locally weighted regression. Minimum jerk trajectories are often used in robotics because they resemble human reaching trajectories [20]. In simulation we first generated Cartesian minimum jerk trajectories, which were converted into joint space trajectories using standard inverse kinematics (see Fig. 2). In Cartesian space these trajectories correspond to straight lines. 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. The query points were distributed 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.

Using B-splines we estimated the mapping (13), which in this case is the mapping from the last end-effector position to the corresponding joint angles

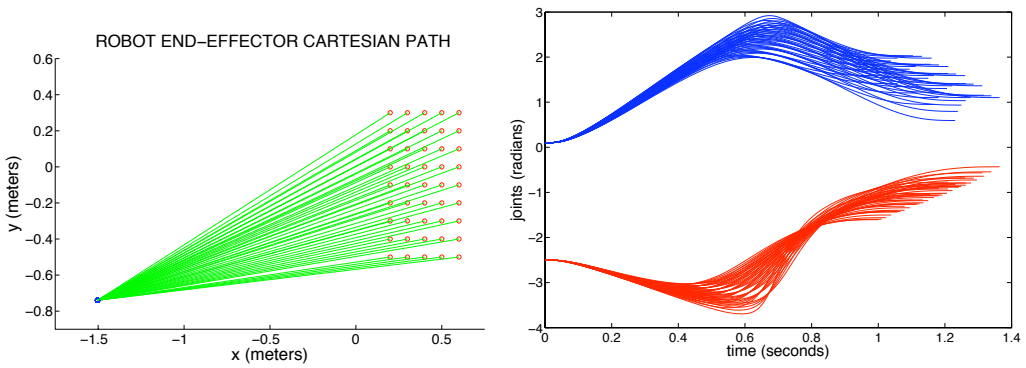

Figure 2: 45 example minimum jerk trajectories in Cartesian space (left) and the associated joint space trajectories (right). In the figure left red circles depict the final reaching positions that were used as query points for locally weighted regression. The sum of limb lengths was 1.31 meters.

and is thus a local approximation for the inverse kinematics. Similarly we estimated the mapping (12) from query points $\mathbf{q}$ to the time duration of the trajectory. In simulation 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. 1 and 2 were calculated by integrating equation system (6) – (8) 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. 1 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. 2, and in the reduced area enclosed by query

Table 1: Errors in reaching movements (in centimeters and degrees, respectively) synthesized by locally weighted regression.

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

Table 2: Errors in reaching movements (in centimeters and degrees, respectively) generated by a single dynamic movement primitive.

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

points situated at least one query point away from the boundary points. As expected, the errors are slightly 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 (13). 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. The accuracy is sufficient to realize high-precision reaching movements for grasping.

Tab. (2) 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 (12). If we allowed the time to flow and continued to integrate the underlying differential equations, the motion would continue and the desired

Table 3: Errors in the synthesized throws represented by splines (in centimeters)

	Joint positions without velocities		Joint positions and velocities		Grid size (centimeters)
Training area	Full	Reduced	Full	Reduced	
Average error	2.41	1.54	2.72	1.75	50×50
Max. error	13.35	6.17	12.57	7.08	50×50

Table 4: Errors in the synthesized throws represented by DMPs (in centimeters)

	Adams-Bashforth-Moulton integration		Euler integration		Grid size (centimeters)
Training area	Full	Reduced	Full	Reduced	
Average error	2.76	2.16	5.16	4.89	50×50
Max. error	16.03	14.76	15.61	11.95	50×50

position would eventually be reached, but with a certain delay.

4.2. Ball throwing

As a second test example we considered a task were the goal is less directly linked to movements than in the case of reaching. We studied the problem of throwing a ball into a basket, which is a dynamic task dependent not only on the positional part of the movement, but also on velocities. It can easily be shown that the trajectory of the ball after the release is fully specified by the position and velocity at the release point

$$x = x_0 + v_0 t \cos(\alpha),\ y = y_0 + v_0 t \sin(\alpha) - \frac{gt^2}{2}, \tag{20}$$

where (x_0, y_0) is the release point, v_0 is the linear velocity of the ball at release time and α is the initial angle of the throw. We considered the problem where the target basket is placed in xy-plane. Since a robot can turn towards the basket, solving this problem allows the robot to throw the ball to any position in space. The understanding of the physics of the task allows us to compare the movement generalization results with an ideal system.

The basket positions, i. e. the positions where the ball needs to land, were used as query points. They were uniformly distributed with spacing of 0.5 meters within a rectangular area with corners at $(1.2, 0.1)$ and $(5.2, 2.1)$

meters. Example movements with proper position and velocity at the release time for the given basket position were generated. Instead of the complete time duration we used the timing at the release point when estimating function (12). Function (13) encoded the mapping from the basket position to the final configuration of the discrete movement trajectory, which was encoded by a DMP.

As can be seen in Tab. 4, the average accuracy of the throws generated by DMPs is comparable to the average accuracy of the throws generated by splines. However, this is the case only if a more advanced integration technique such as Adams-Bashforth-Moulton method [21] is used to calculate the positions, velocities, and accelerations on the trajectory encoded by the Eq. (6) – (8). Average errors are significantly larger when a simpler Euler's method is used for integration. On the other hand, maximum error was larger with both methods, especially in the reduced area. This leads us to believe that while DMPs approximate the desired trajectories and their dynamics with an accuracy comparable to splines, care must be taken to achieve a good enough accuracy for tasks that require high precision such as ball throwing. The representation with a single DMP is not precise enough to hit the target, therefore the results obtained with this method are omitted.

5. Summary and Conclusion

The movement generalization approach proposed in this paper is realized using locally weighted regression that defers learning to the execution time. Unlike many other approaches that were proposed to modify DMPs with respect to the external perceptual input and that require an expert to modify the underlying system of differential equations, our approach generates new movements directly from the data with percepts used as query points. 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 on-line using current perceptual input.

The experimental results show that by using DMPs and locally weighted regression, we can approximate a space of smooth movements, e. g. minimum jerk reaching movements, with high accuracy. While locally weighted regression can be applied to other representations such as B-splines [8], it has been shown that in the execution phase, dynamic systems have many advantages over nonautonomous representations including splines. This is due to the ease with which the learned movements can be adapted to unforeseen perturbations [13, 15]. The proposed approach thus enables the robot to execute 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, such as in the case of purely feedforward movements that constitute throwing, splines might be preferable because they do not suffer from problems associated with the integration of differential equations.

As described in Section 3.1, the generalization of movements makes sense only for problems with example movements that transition smoothly as a function of query points. This is, however, not always the case. Consider for example reaching movements that need to avoid an obstacle before arriving to the final configuration. If there are two sets of example movements, 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.

Acknowledgment: This research was supported in part by the EU Cognitive Systems project PACO-PLUS (FP6-027657).

References

[1] M. Lungarella, G. Metta, R. Pfeifer, G. Sandini, Developmental robotics: a survey, Connection Science 15 (4) (2003) 151–190.

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

[3] M. Riley, A. Ude, 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, Pittsburgh, Pennsylvania, 2000, pp. 35–42.

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

[5] M. Ruchanurucks, S. Nakaoka, S. Kudoh, K. Ikeuchi, Humanoid robot motion generation with sequential physical constraints, in: Proc. IEEE Int. Conf. Robotics and Automation, Orlando, Florida, 2006, pp. 2649–2654.

[6] H. Miyamoto, S. Schaal, F. Gandolfo, H. Gomi, Y. Koike, R. Osu, E. Nakano, Y. Wada, M. Kawato, A kendama learning robot based on bi-directional theory, Neural Networks 9 (8) (1996) 1281–1302.

[7] M. Riley, A. Ude, C. G. Atkeson, G. Cheng, Coaching: An approach to efficiently and intuitively create humanoid robot behaviors, in: Proc. IEEE-RAS Int. Conf. Humanoid Robots, Genoa, Italy, 2006, pp. 567–574.

[8] A. Ude, M. Riley, B. Nemec, A. Kos, T. Asfour, G. Cheng, Synthesizing goal-directed actions from a library of example movements, in: Proc. IEEE-RAS Int. Conf. Humanoid Robots, Pittsburgh, Pennsylvania, 2007.

[9] T. Asfour, F. Gyarfas, P. Azad, R. Dilmann, Imitation learning of dual-arm manipulation tasks in humanoid robots, International Journal of Humanoid Robotics 5 (2) (2008) 183–202.

[10] A. Billard, S. Calinon, F. Guenter, Discriminative and adaptive imitation in uni-manual and bi-manual tasks, Robotics and Autonomous Systems 54 (2006) 370–384.

[11] T. Inamura, I. Toshima, H. Tanie, Y. Nakamura, Embodied symbol emergence based on mimesis theory, Int. J. Robotics Research 23 (4-5) (2004) 363–377.

[12] A. J. Ijspeert, J. Nakanishi, 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.

[13] A. J. Ijspeert, J. Nakanishi, S. Schaal, Learning attractor landscapes for learning motor primitives, in: S. Becker, S. Thrun, K. Obermayer

(Eds.), Advances in Neural Information Processing Systems 15, MIT Press, Cambridge, Mass., 2003, pp. 1547–1554.

[14] S. Schaal, J. Peters, J. Nakanishi, A. Ijspeert, Learning movement primitives, in: P. Dario, R. Chatila (Eds.), Robotics Research: The Eleventh International Symposium, Springer, Berlin, Heidelberg, 2005, pp. 561–572.

[15] S. Schaal, P. Mohajerian, A. Ijspeert, Dynamics systems vs. optimal control – a unifying view, Progress in Brain Research 165 (6) (2007) 425–445.

[16] P. Pastor, H. Hoffmann, T. Asfour, S. Schaal, Learning and generalization of motor skills by learning from demonstration, in: Proc. IEEE Int. Conf. Robotics and Automation, Kobe, Japan, 2009 (to appear).

[17] D.-H. Park, H. Hoffmann, P. Pastor, S. Schaal, Movement reproduction and obstacle avoidance with dynamic movement primitives and potential fields, in: Proc. IEEE-RAS Int. Conf. Humanoid Robots, Daejeon, Korea, 2008.

[18] A. Gams, A. J. Ijspeert, S. Schaal, J. Lenarčič, On-line learning and modulation of periodic movements with nonlinear dynamical systems, Submitted to Autonomous Robots (2009).

[19] C. G. Atkeson, A. W. Moore, S. Schaal, Locally weighted learning, AI Review 11 (1997) 11–73.

[20] T. Flash, N. Hogan, The coordination of arm movements: an experimentally confirmed mathematical model, The Journal of Neuroscience 5 (7) (1985) 1688–1703.

[21] J. D. Hoffman, Numerical Methods for Engineers and Scientists, 2nd Edition, CRC Press, 2001.

Learning and Generalization of Motor Skills by Learning from Demonstration

Peter Pastor, Heiko Hoffmann, Tamim Asfour, and Stefan Schaal

***Abstract*— We provide a general approach for learning robotic motor skills from human demonstration. To represent an observed movement, a non-linear differential equation is learned such that it reproduces this movement. Based on this representation, we build a library of movements by labeling each recorded movement according to task and context (e.g., grasping, placing, and releasing). Our differential equation is formulated such that generalization can be achieved simply by adapting a start and a goal parameter in the equation to the desired position values of a movement. For object manipulation, we present how our framework extends to the control of gripper orientation and finger position. The feasibility of our approach is demonstrated in simulation as well as on a real robot. The robot learned a pick-and-place operation and a water-serving task and could generalize these tasks to novel situations.**

I. INTRODUCTION

Humanoid robots assisting humans can become widespread only if the humanoids are easy to program. Easy programming might be achieved through learning from demonstration [1], [2]. A human movement is recorded and later reproduced by a robot. Three challenges need to be mastered for this imitation: the correspondence problem, generalization, and robustness against perturbation. The correspondence problem means that links and joints between human and robot may not match. Generalization is required because we cannot demonstrate every single movement that the robot is supposed to make. Learning by demonstration is feasible only if a demonstrated movement can be generalized to other contexts, like different goal positions. Finally, we need robustness against perturbation. Replaying exactly an observed movement is unrealistic in a dynamic environment, in which obstacles may appear suddenly.

To address these issues, we present a model that is based on the dynamic movement primitive (DMP) framework (see [3], [4]). In this framework, any recorded movement can be represented with a set of differential equations. Representing a movement with a differential equation has the advantage that a perturbance can be automatically corrected for by the dynamics of the system; this behavior addresses the above mentioned flexibility. Furthermore, the equations are formulated in a way that adaptation to a new goal is achieved by simply changing a goal parameter. This characteristic allows generalization. Here, we will present a new version of the dynamic equations that overcomes numerical problems with changing the goal parameter that occurred in the previous formulation [5].

We will use the dynamic movement primitives to represent a movement trajectory in end-effector space; thus, we address the above-mentioned correspondence problem. For object manipulation – here, grasping and placing – besides the end-effector position, we also need to control the orientation of the gripper and the position of the fingers. The DMP framework allows to combine the end-effector motion with any further degree-of-freedom (DOF); thus, adding gripper orientation in quaternion notation and finger position is straight-forward. In our robot demonstration, we use standard resolved motion rate inverse kinematics to map end-effector position and gripper orientation onto the appropriate joint angles [6].

To deal with complex motion, the above framework can be used to build a library of movements primitives out of which complex motion can be composed by sequencing. For example, the library may contain a grasping, placing, and releasing motion. Each of these movements, which was recorded from human demonstration, is represented by a differential equation, and labeled accordingly. For moving an object on a table, a grasping-placing-releasing sequence is required, and the corresponding primitives are recalled from the library. Due to the generalization ability of each dynamic movement primitive, an object may be placed between two arbitrary positions on the table based solely on the three demonstrated movements.

In the remainder of this article, we explain in Section II the dynamic movement primitive framework and present the new modified form. In Section III we emphasize the generation of a library of movements. In Section IV we present an application of the framework on a simulated as well as on a real robot arm. In Section V we conclude this approach and provide an outlook for future work.

II. DYNAMIC SYSTEMS FOR MOVEMENT GENERATION

This section briefly describes the dynamic movement primitive framework and presents a modification to allow adaptation to a new goal position in a more robust and human-like way.

This research was supported in part by National Science Foundation grants ECS-0325383, IIS-0312802, IIS-0082995, ECS-0326095, ANI-0224419, the DARPA program on Learning Locomotion, the ATR Computational Neuroscience Laboratories, DFG grant HO-3887/1-1, and partially by the EU Cognitive Systems projects PACO-PLUS (FP6-027657) and GRASP (FP7-215821).

Peter Pastor, Heiko Hoffmann, and Stefan Schaal are with the University of Southern California, Los Angeles, USA. Tamim Asfour is with the University of Karlsruhe, Germany. Email: pastorsa@usc.edu, heiko@clmc.usc.edu, asfour@ira.uka.de, sschaal@usc.edu

A. Dynamic Movement Primitives

Dynamic movement primitives can be used to generate discrete and rhythmic movements. Here, we focus on discrete movements. A one dimensional movement is generated by integrating the following set of differential equations[1], which can be interpreted as a linear spring system perturbed by an external forcing term:

$$\tau \dot{v} = K(g-x) - Dv + (g-x_0)f \quad (1)$$

$$\tau \dot{x} = v \ , \quad (2)$$

where x and v are position and velocity of the system; x_0 and g are the start and goal position; τ is a temporal scaling factor; K acts like a spring constant; the damping term D is chosen such that the system is critically damped, and f is a non-linear function which can be learned to allow the generation of arbitrarily complex movements. This first set of equations is referred to as a transformation system. The non-linear function is defined as

$$f(\theta) = \frac{\sum_i w_i \psi_i(\theta)\theta}{\sum_i \psi_i(\theta)} \ , \quad (3)$$

where $\psi_i(\theta) = \exp(-h_i(\theta - c_i)^2)$ are Gaussian basis functions, with center c_i and width h_i, and w_i are adjustable weights. The function f does not directly depend on time; instead, it depends on a phase variable θ, which monotonically changes from 1 towards 0 during a movement and is obtained by the equation

$$\tau \dot{\theta} = -\alpha \, \theta \ , \quad (4)$$

where α is a pre-defined constant. This last differential equation is referred to as canonical system. These sets of equations have some favorable characteristics:

- Convergence to the goal g is guaranteed (for bounded weights) since $f(\theta)$ vanishes at the end of a movement.
- The weights w_i can be learned to generate any desired smooth trajectory.
- The equations are spatial and temporal invariant, i.e., movements are self-similar for a change in goal, start point, and temporal scaling without a need to change the weights w_i.
- The formulation generates movements which are robust against perturbation due to the inherent attractor dynamics of the equations.

To learn a movement from demonstration, first, a movement $x(t)$ is recorded and its derivatives $v(t)$ and $\dot{v}(t)$ are computed for each time step $t = 0, \ldots, T$. Second, the canonical system is integrated, i.e., $\theta(t)$ is computed for an appropriately adjusted temporal scaling τ. Using these arrays, $f_{\text{target}}(\theta)$ is computed based on (1) according to

$$f_{\text{target}}(\theta) = \frac{-K(g-x) + Dv + \tau \dot{v}}{g - x_0} \ , \quad (5)$$

where x_0 and g are set to $x(0)$ and $x(T)$, respectively. Thus, finding the weights w_i in (3) that minimize the error criterion $J = \sum_\theta \left(f_{\text{target}}(\theta) - f(\theta)\right)^2$ is a linear regression problem, which can be solved efficiently.

A movement plan is generated by reusing the weights w_i, specifying a desired start x_0 and goal g, setting $\theta = 1$, and integrating the canonical system, i.e. evaluating $\theta(t)$. As illustrated in Fig. 1, the obtained phase variable then drives the non-linear function f which in turn perturbs the linear spring-damper system to compute the desired attractor landscape.

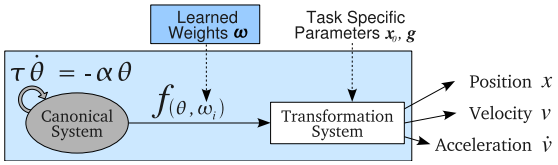

Fig. 1. Sketch of a one dimensional DMP: the canonical system drives the nonlinear function f which perturbs the transformation system.

B. Generalization to New Goals

In this section, we describe how to adapt the movement to a new goal position by changing the goal parameter g. The original DMP formulation has three drawbacks: first, if start and goal position, x_0 and g, of a movement are the same, then the non-linear term in (1) cannot drive the system away from its initial state; thus, the system will remain at x_0. Second, the scaling of f with $g - x_0$ is problematic if $g - x_0$ is close to zero; here, a small change in g may lead to huge accelerations, which can break the limits of the robot. Third, whenever a movement adapts to a new goal g_{new} such that $(g_{\text{new}} - x_0)$ changes its sign compared to $(g_{\text{original}} - x_0)$ the resulting generalization is mirrored. As an example from our experiments, a placing movement on a table has start and goal positions with about the same height; thus, the original DMP formulation is not suitable for this kind of movement adaptation.

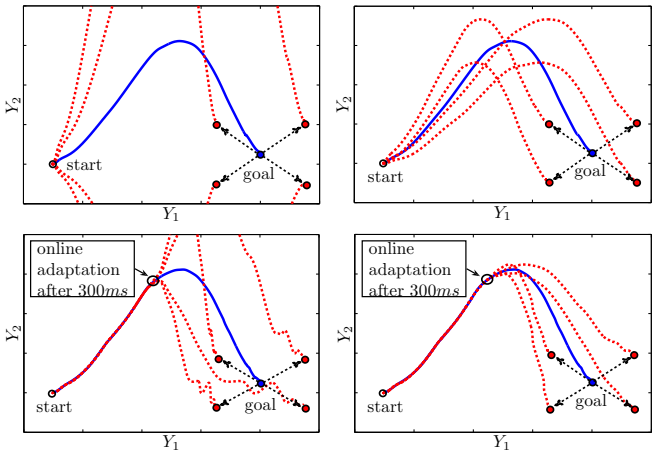

Fig. 2. Comparison of goal-changing results between old (Left) and new (Right) DMP formulation in operational space (Y_1, Y_2) with one transformation system for each dimension. The same original movement (solid line) and goals are used for both formulations. The dashed lines show the result of changing the goal before movement onset (Top) and during the movement (Bottom).

[1]We use a different notation as in [3] to highlight the spring-like character of these equations.

Here, we present a modified form of the DMPs that cures these problems (see Fig. 2), while keeping the same favorable properties, as mentioned above. We replace the transformation system by the following equations [5]:

$$\tau \dot{v} = K(g-x) - Dv - K(g-x_0)\theta + Kf(\theta) \quad (6)$$
$$\tau \dot{x} = v \ , \quad (7)$$

where the non-linear function $f(\theta)$ is defined as before. We use the same canonical system as in (4). An important difference from (1) is that the non-linear function is not multiplied any more by $(g-x_0)$. The third term $K(g-x_0)\theta$ is required to avoid jumps at the beginning of a movement. Learning and propagating DMPs is achieved with the same procedure as before, except that the target function $f_{\text{target}}(\theta)$ is computed according to

$$f_{\text{target}}(\theta) = \frac{\tau \dot{v} + D\, v}{K} - (g-x) + (g-x_0)\,\theta \ . \quad (8)$$

The trajectories generated by this new formulation for different g values are shown in Fig. 2. In our simulation and robot experiments we use this new formulation.

C. Obstacle Avoidance

A major feature of using dynamic systems for movement representation is robustness against perturbation [3]. Here, we exploit this property for obstacle avoidance [7] by adding a coupling term $p(\mathbf{x}, \mathbf{v})$ to the differential equations of motion

$$\tau \dot{\mathbf{v}} = \mathbf{K}(\mathbf{g}-\mathbf{x}) - \mathbf{D}\mathbf{v} - \mathbf{K}(\mathbf{g}-\mathbf{x}_0)\theta + \mathbf{K}\mathbf{f}(\theta) + \mathbf{p}(\mathbf{x}, \mathbf{v}) \ . \quad (9)$$

We describe obstacle avoidance in 3D end-effector space, therefore the scalars $x, v, \dot{v}$ turn into vectors $\mathbf{x}, \mathbf{v}, \dot{\mathbf{v}}$ and the scalars K, D became positive definite matrices $\mathbf{K}, \mathbf{D}$. For the experiment in this paper we used the coupling term

$$\mathbf{p}(\mathbf{x}, \mathbf{v}) = \gamma\, \mathbf{R}\, \mathbf{v}\, \varphi \exp(-c\varphi) \ , \quad (10)$$

where $\mathbf{R}$ is a rotational matrix with axis $\mathbf{r} = (\mathbf{x}-\mathbf{o}) \times \mathbf{v}$ and angle of rotation of $\pi/2$; the vector $\mathbf{o}$ is the position of the obstacle, γ and c are constant, and φ is the angle between the direction of the end-effector towards the obstacle and the end-effector's velocity vector $\mathbf{v}$ relative to the obstacle. The expression (10) is derived from [8] and empirically matches human obstacle avoidance. In the robot experiment we used $\gamma = 1000$ and $c = 20$.

III. BUILDING A LIBRARY OF MOVEMENTS

This section briefly motivates the concept of a library of movements and their application in object manipulation tasks.

A. Motion Library Generation

Learning DMPs only requires the user to demonstrate characteristic movements. These DMPs form a set of basic units of action [1]. For movement reproduction only a simple high level command - to choose a primitive (or a sequence of them) and set its task specific parameters - is required.

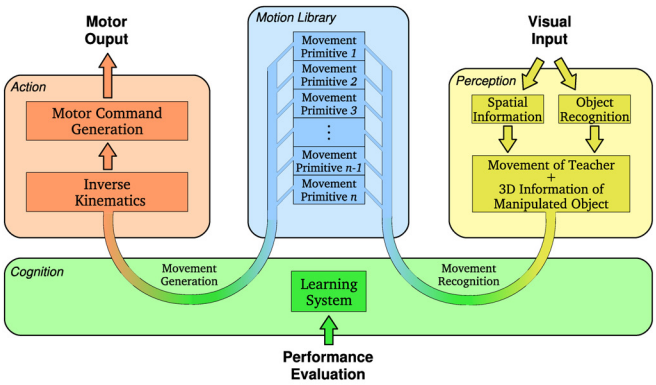

Fig. 3. Conceptual sketch of an imitation learning system (adapted from [1]). The components of perception (yellow) transform visual information into spatial and object information. The components of action (red) generate motor output. Interaction between them is achieved using a common motion library (blue). Learning (green) improves the mapping between perceived actions and primitives contained in the motion library for movement recognition and selection of the most appropriate primitive for movement generation.

Moreover, adaption to new situations is accomplished by adjusting the start $\mathbf{x}_0$, the goal $\mathbf{g}$, and the movement duration τ. Thus, a collection of primitives referred to as *motion library* enables a system to generate a wide range of movements. On the other side, such a motion library can be employed to facilitate movement recognition in that observed movements can be compared to the pre-learned ones [3]. If no existing primitive is a good match for the demonstrated behavior, a new one is created (learned) and added to the system's movement repertoire (see Fig. 3). This makes the presented formulation suited for imitation learning.

B. Attaching Semantic

As for imitation learning with DMPs a low-level approach, namely imitation of trajectories [2], was chosen, additional information is needed by the system to successfully perform object manipulation tasks. For a pick-and-place operation for example the system has to select a proper sequence of movement primitives, that is, first a grasping, then a placing and finally a releasing primitive. Therefore, it is necessary to attach additional information to each primitive movement which facilitates this selection. Moreover, once a library of movement primitives is acquired, it is desirable to have the system be able to find sequences of primitive movements that accomplish further tasks. Traditional artificial intelligence planning algorithms tackle this problem by formalizing the

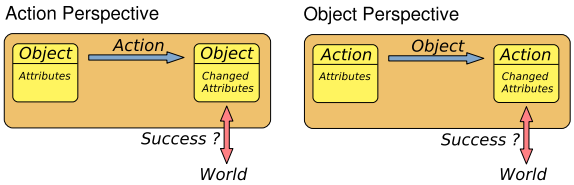

Fig. 4. Objects are defined through actions that can be performed on them (Left), e.g. a cup is represented as a thing which can be used to drink water from. On the other side, actions are defined through objects (Right), e.g. the way of grasping an object depends on the object - a can requires a different grip than a pen.

domain scenario. In particular, they define a set of operators with pre- and post-conditions and search for a sequence of them which transfers the world from its initial state to the goal state. The post-conditions provides information about the change in the world, whereas the preconditions ensure that the plan is executable. Thus, such algorithms are based on discrete symbolic representations of object and action, rather than the low-level continuous details of action execution.

A link between the low-level continuous control representation (as typical in robotic applications) and high-level formal description of actions and their impact on objects (as necessary for planning) has been, for example, formalized by the concept of Object-Action Complexes [9], [10]. This concept proposes that objects and actions are inseparably intertwined (see Fig. 4).

C. Combination of Movement Primitives

The ability to combine movement primitives to generate more complex movements is a prerequisite for the concept of a motion library. Here, we show how the presented framework provides this ability.

It is straight forward to start executing a DMP after the preceding DMP has been executed completely, since the boundary conditions of any DMP are zero velocity and acceleration. However, DMPs can also be sequenced such that complete stops of the movement system are avoided (see Fig. 5). This is achieved by starting the execution of the successive DMP before the preceding DMP has finished. In this case, the velocities and accelerations of the movement system between two successive DMPs are not zero. Jumps in the acceleration signal are avoided by properly initializing the succeeding DMP with the velocities and positions of its predecessor ($\mathbf{v}_{\text{pred}} \rightarrow \mathbf{v}_{\text{succ}}$ and $\mathbf{x}_{\text{pred}} \rightarrow \mathbf{x}_{\text{succ}}$).

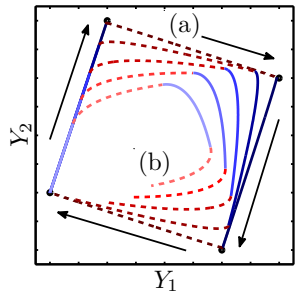

Fig. 5. Chaining of a single minimum jerk movement primitive generalized to four goals (black dots) resulting in a square like movement (a). The movement generated by the DMPs are drawn alternating with blue solid and red dashed lines to indicate the transition between two successive DMPs. The movement direction is indicated by the arcs. The remaining movements (b) result from using different switching times (lighter color indicates earlier switching time).

IV. EXPERIMENT

The following Section describes how we applied the presented framework of DMPs on the Sarcos Slave arm to accomplish object manipulation tasks, such as grasping and placing. As experimental platform we used a seven DOF anthropomorphic robot arm (see Fig. 6) equipped with a three DOF end-effector.

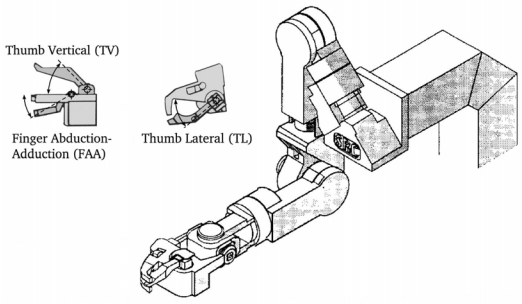

Fig. 6. Sketch of the Sarcos Slave arm, a seven DOF anthropomorphic robot arm with a three DOF end-effector.

A. Learning DMPs from Demonstration

Learning DMPs from demonstration is achieved by regarding each DOF separately and employing for each of them an individual transformation system. Thus, each DMP is setup with a total of ten transformation systems to encode each kinematic variable. In particular, the involved variables are the end-effector's position (x, y, z) in Cartesian space, the end-effector's orientation (q_0, q_1, q_2, q_3) in quaternion space, and finger position $(\theta_{\text{TL}}, \theta_{\text{TV}}, \theta_{\text{FAA}})$ in joint space. Each of them serve as a separate learning signal, regardless of the underlying physical interpretation. However, to ensure the unit length of the quaternion $\mathbf{q}$, a post-normalization step is incorporated. The setup is illustrated in Fig. 7, note, a single DMP encodes movements in three different coordinate frames simultaneously.

To record a set of movements, we used a 10 DOF exoskeleton robot arm, as shown in Fig. 8. Visual observation and appropriate processing to obtain the task variables would be possible, too, but was avoided as this perceptual component is currently not the focus of our research.

The end-effector position and orientation are recorded at 480 Hz. The corresponding trajectories for the finger move-

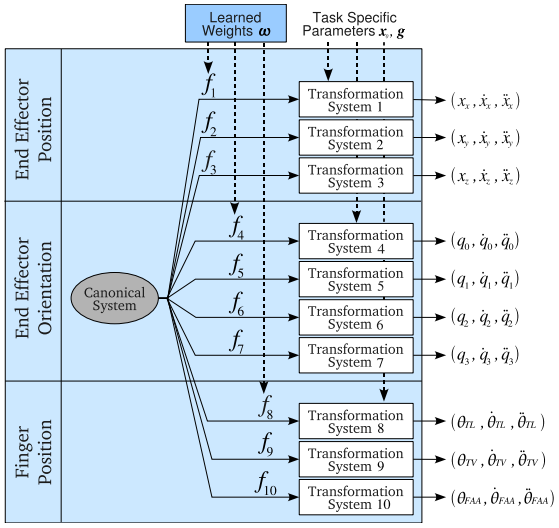

Fig. 7. Sketch of the 10 dimensional DMP used to generate movement plans for the Sarcos Slave arm.

Fig. 8. Sarcos Master arm used to record a human trajectory in end-effector space. Here, the subject demonstrates a pouring movement which after learning the DMP enabled a robot to pour water into several cups (see Fig. 12).

ments are generated afterwards accordingly: for a grasping movement, for example, a trajectory was composed out of two minimum jerk movements for opening and closing the gripper. The corresponding velocities and accelerations for all DOF were computed numerically by differentiating the position signal.

These signals served as input into the supervised learning procedure described in II-A. For each demonstrated movement a separate DMP was learned and added to the motion library.

B. Movement Generation

To generate a movement plan, a DMP is setup with the task specific parameters, i.e., the start $\mathbf{x_0}$ and the goal $\mathbf{g}$. In our DMP setup (see Fig. 7), these are the end-effector position, end-effector orientation, and the finger joint configuration. The start $\mathbf{x_0}$ of a movement is set to the current state of the robot arm. The goal $\mathbf{g}$ is set according to the context of the movement. For a grasping movement, the goal position (x, y, z) is set to the position of the grasped object and the grasping width is set according to the object's size. However, finding an appropriate goal orientation is not straight forward, as the end-effector orientation needs to be adapted to the characteristic approach curve of the movement. Approaching the object from the front results in a different final posture as approaching it from the side. In case of a grasping movement, we developed a method to automatically determine the final orientation by propagating the DMP to generate the Cartesian space trajectory and averaging over the velocity vectors to compute the approach direction at the end of the movement. For other movements, like placing and releasing, we set the end-effector orientation to the orientation recorded from human demonstration. Finally, we use τ to determine the duration of each movement. In simulation we demonstrate the reproduction and generalization of the demonstrated movements. Our simulated robot arm has the same kinematic and dynamic properties as the Sarcos Slave arm. The reproduction of grasping and placing are show in Fig. 9. The generalization of these movements to new targets is shown in Fig. 10.

C. Task Space Control

To execute DMPs on the robot we used a velocity based inverse kinematics controller as described in [11], [6]. Thus, the task space reference velocities $\dot{\boldsymbol{x}}_r$ are transformed into the reference joint space velocities $\dot{\boldsymbol{\theta}}_r$ (see Fig. 11). The reference joint position $\boldsymbol{\theta}_r$ and acceleration $\ddot{\boldsymbol{\theta}}_r$ are obtained

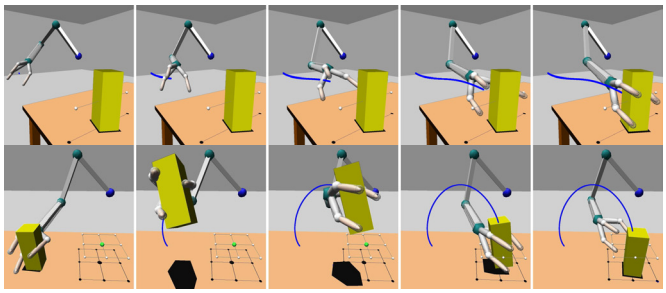

Fig. 9. Snapshots of the *SL* Simulator showing a simulation of the Sarcos Slave arm performing a grasping (Top) and a placing movement (Bottom).

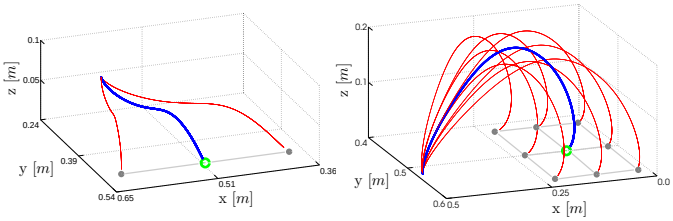

Fig. 10. The desired trajectories (blue lines) from the movements shown in Fig. 9 adapted to new goals (red lines) indicated by the grid.

by numerical integration and differentiation of the reference joint velocities $\dot{\boldsymbol{x}}_r$. The desired orientation, given by the DMP as unit quaternions, is controlled using quaternion feedback as described in [12], [6].

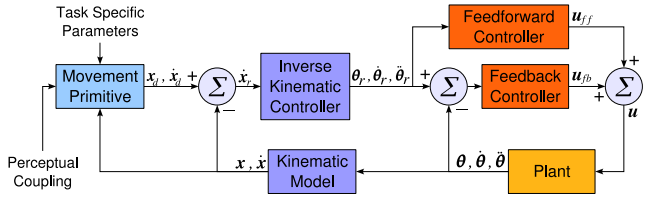

Fig. 11. DMP control diagram: the desired task space positions and velocities are $\boldsymbol{x_d}$, $\dot{\boldsymbol{x}}_{\boldsymbol{d}}$, the reference task space velocity commands are $\dot{\boldsymbol{x}}_{\boldsymbol{r}}$, the reference joint positions, joint velocities, and joint accelerations are $\boldsymbol{\theta}_r$, $\dot{\boldsymbol{\theta}}_r$, and $\ddot{\boldsymbol{\theta}}_r$.

The reference joint position, velocities and acceleration are transformed into appropriate torque commands $\boldsymbol{u}$ using a feed-forward and a feedback component. The feed-forward component estimates the corresponding nominal torques to compensate for all interactions between the joints, while the feedback component realizes a PD controller.

D. Robot Experiment

We demonstrate the utility of our framework in a robot demonstration of water-serving (see Fig. 12). First, a human demonstrator performed a grasping, pouring, retreating bottle, and releasing movement as illustrated in Fig. 8. Second, the robot learned these movements and added them to the motion library. Third, a bottle of water and three cups were placed on the table. Fourth, an appropriate sequence of movement primitives were chosen manually. Fifth, each DMP were setup with corresponding goal g. Finally, the robot executed the sequence of movements and generalized

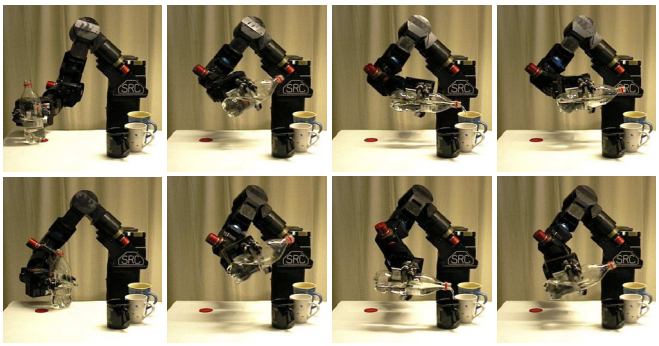

Fig. 12. Movement reproduction and generalization to new goal with the Sarcos Slave Arm. The top row shows the reproduction of a demonstrated pouring movement in Fig. 8, and the bottom row shows the result of changing the goal variable.

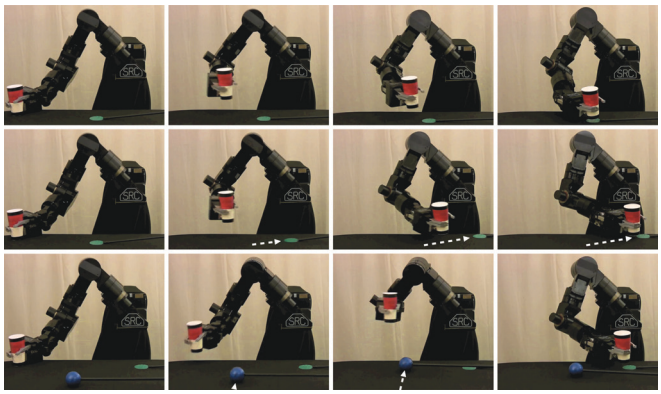

Fig. 13. Sarcos Slave arm placing a red cup on a green coaster. The first row shows the placing movement on a fixed goal. The second row shows the resulting movement as the goal changes (white dashed arc) after movement onset. The third row shows the resulting movement as a blue ball-like obstacle interferes with the placing movement.

to different cup position simply through changing the goal g of the pouring movement.

To demonstrate the framework's ability to adapt online to new goals as well as avoid obstacles, we extended the experimental setup with a stereo camera system. We used a color based vision system to visually extract the goal position as well as the position of the obstacle. The task was to grasp a red cup and place it on a green coaster, which changes its position after movement onset, while avoiding a blue ball-like obstacle (see Fig. 13). To accomplish this task a similar procedure was used as before. Except, this time, the Cartesian goal of the grasping movement was set to the position of the red cup and the goal of the placing movement was set to the green coaster. The goal orientation for the grasping movement were set automatically as described in Section IV-B, whereas the orientation of the placing and releasing were adopted from demonstration. This setup allows us to demonstrate the framework's ability to generalize the grasping movement by placing the red cup on different initial positions. Our robot could adapt movements to goals which change their position during the robot's movement. Additionally, movement trajectories were automatically adapted to avoid moving obstacles (see Fig. 13 and video supplement).

V. CONCLUSIONS AND FUTURE WORK

This paper extended the framework of dynamic movement primitives to action sequences that allow object manipulation. We suggested several improvements of the original movement primitive framework, and added semantic information to movement primitives, such that they can code object oriented action. We demonstrated the feasibility of our approach in an imitation learning setting, where a robot learned a water-serving and a pick-and-place task from human demonstration, and could generalize this task to novel situations.

The approach is not restricted to the presented experimental platform. Any type of motion capture system that is capable of extracting the end-effector's position and orientation can substitute the Sarcos Master arm and any manipulator that is able to track a reference trajectory in task space can substitute the Sarcos Slave arm.

Future work will significantly extend the movement library such that a rich movement repertoire can be represented. Furthermore, work will focus on associating objects with actions (similar to [10]) to enable planning of action sequences. Finally, we will apply this extended framework on a humanoid robot.

REFERENCES

[1] S. Schaal, "Is imitation learning the route to humanoid robots?" in *Trends in Cognitive Sciences*, 1999.

[2] A. Billard and R. Siegwart, "Robot Learning from Demonstration," *Robotics and Autonomous Systems*, vol. 47, pp. 65–67, 2004.

[3] A. J. Ijspeert, J. Nakanishi, and S. Schaal, "Movement Imitation with Nonlinear Dynamical Systems in Humanoid Robots," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2002.

[4] S. Schaal, A. J. Ijspeert, and A. Billard, "Computational Approaches to Motor Learning by Imitation," in *Philosophical Transaction of the Royal Society of London: Series B, Biological Sciences*, 2003.

[5] H. Hoffmann, P. Pastor, and S. Schaal, "Dynamic movement primitives for movement generation motivated by convergent force fields in frog," in *Fourth International Symposium on Adaptive Motion of Animals and Machines*, R. Ritzmann and R. Quinn, Eds., Case Western Reserve University, Cleveland, OH, 2008.

[6] J. Nakanishi, R. Cory, M. Mistry, J. Peters, and S. Schaal, "Operational space control: a theoretical and emprical comparison," *International Journal of Robotics Research*, vol. 27, pp. 737–757, 2008.

[7] D. Park, H. Hoffmann, P. P., and S. Schaal, "Movement reproduction and obstacle avoidance with dynamic movement primitives and potential fields," in *IEEE International Conference on Humanoid Robotics*, J. H. Oh, Ed., Daejeon, Korea, submitted.

[8] B. R. Fajen and W. H. Warren, "Behavioral dynamics of steering, obstacle avoidance, and route selection," *Journal of Experimental Psychology: Human Perception and Performance*, vol. 29, no. 2, pp. 343–362, 2003.

[9] T. Asfour, K. Welke, A. Ude, P. Azad, and R. Dillmann, "Perceiving objects and movements to generate actions on a humanoid robot," in *Unifying Perspectives in Computational and Robot Vision (Lecture Notes in Electrical Engineering)*, D. Kragic and V. Kyrki, Eds. Springer Publishing Company, Incorporated, 2008.

[10] C. Geib, K. Mourao, R. Petrick, N. Pugeault, M. Steedman, N. Krueger, and F. Woergoetter, "Object Action Complexes as an Interface for Planning and Robot Control," in *Proceedings of the IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, 2006.

[11] J. Nakanishi, M. Mistry, J. Peters, and S. Schaal, "Experimental evaluation of task space position/orientation control towards compliant control for humanoid robots," in *Proceedings of the IEEE International Conference on Intelligent Robotics Systems*, 2007, pp. 1–8.

[12] J. S. Yuan, "Closed-loop manipulator control using quaternion feedback," *IEEE Journal of Robotics and Automation*, vol. 4, no. 4, pp. 434–440, 1988.