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 the Sarcos dextrous robot arm. The robot learned a pick-and-place operation and a water-serving task and could generalize these tasks to novel situations.**

I. INTRODUCTION

Anthropomorphic robots assisting humans can become widespread only if these robots are easy to program. Easy programming might be achieved through learning from demonstration [1], [2], [3]. A human movement is recorded and later reproduced by a robot. Three challenges need to be mastered for this imitation: the correspondence problem [4], 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 [5], [6]. 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 (robustness against perturbation). 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 [7].

We represent a movement trajectory in end-effector space to address the 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 additional degree-of-freedom (DOF); thus, adding gripper orientation in quaternion notation and finger posture is straight-forward. In our robot demonstration, we use resolved motion rate inverse kinematics to map end-effector position and gripper orientation onto the appropriate joint angles [8].

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 movement. 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. Section III describes building a library of movements. In Section IV, we present an application of the framework on a simulated as well as on a real Sarcos 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, discusses movement generalization to new goals, presents our modified DMP formualtion, and its extension to obstacle avoidance.

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

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 (TH), Germany. Email: pastorsa@usc.edu, heiko@clmc.usc.edu, asfour@ira.uka.de, sschaal@usc.edu

integrating the following set of differential equations[1], which can be interpreted as a linear spring system perturbed by an external forcing term:

$$\tau \dot{v} = K(g-x) - Dv + (g-x_0)\,f \tag{1}$$
$$\tau \dot{x} = v\ , \tag{2}$$

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

$$f(s) = \frac{\sum_i w_i \psi_i(s) s}{\sum_i \psi_i(s)}\ , \tag{3}$$

where $\psi_i(s) = \exp(-h_i(s-c_i)^2)$ 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 s, which monotonically changes from 1 towards 0 during a movement and is obtained by the equation

$$\tau \dot{s} = -\alpha\, s\ , \tag{4}$$

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 $f(s)$ 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., $s(t)$ is computed for an appropriately adjusted temporal scaling τ. Using these arrays, $f_{\text{target}}(s)$ is computed based on (1) according to

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

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 $J = \sum_s \big(f_{\text{target}}(s) - f(s)\big)^2$ 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 $s = 1$, and integrating the canonical system, i.e. evaluating $s(t)$. 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.

[1]We use a different notation as in [5] to highlight the spring-like character of these equations.

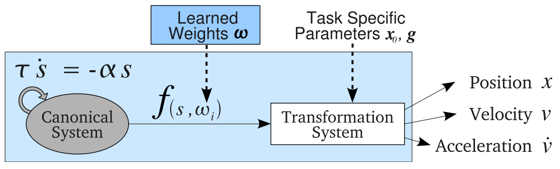
Learned Weights $\boldsymbol{\omega}$
Task Specific Parameters $\boldsymbol{x}_0$, $\boldsymbol{g}$
$\tau\dot{s} = -\alpha s$
Canonical System
$f(s, \omega_i)$
Transformation System
Position x
Velocity v
Acceleration $\dot{v}$

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 unsuitable for adapting this movement to new goal positions.

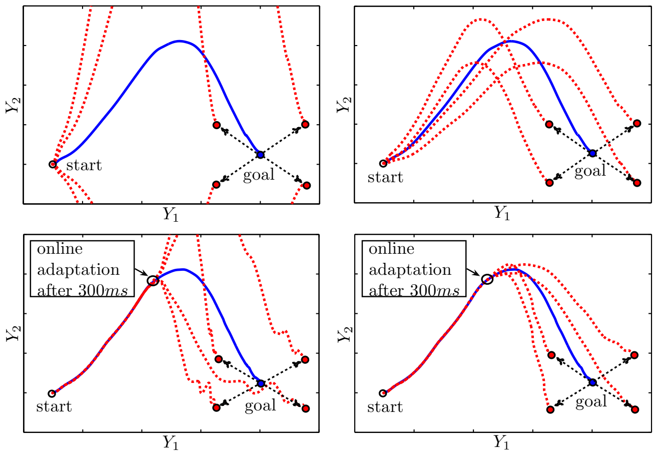

Fig. 2. Comparison of goal-changing results between old (Left) and new (Right) DMP formulation in operational space (Y_1, Y_2) with one transformation system for each dimension. The same original movement (solid line) and goals are used for both formulations. The dashed lines show the result of changing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

C. Modified Dynamic Movement Primitives

Here, we present a modified form of the DMPs that cures these problems (Fig. 2), while keeping the same favorable properties, as mentioned above. We replace the transformation system by the following equations [7]:

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

where the non-linear function $f(s)$ 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 $K(g-x_0)s$ 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 $f_{\text{target}}(s)$ is computed according to

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

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.

D. Obstacle Avoidance

A major feature of using dynamic systems for movement representation is robustness against perturbation [5]. Here, we exploit this property for obstacle avoidance [9] by adding a coupling term $p(\mathbf{x}, \mathbf{v})$ to the differential equations of motion

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

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

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

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 β 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 [7]. The expression (10) is derived from [10] and empirically matches human obstacle avoidance. In the robot experiment we used $\gamma = 1000$ and $\beta = 20$.

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. 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

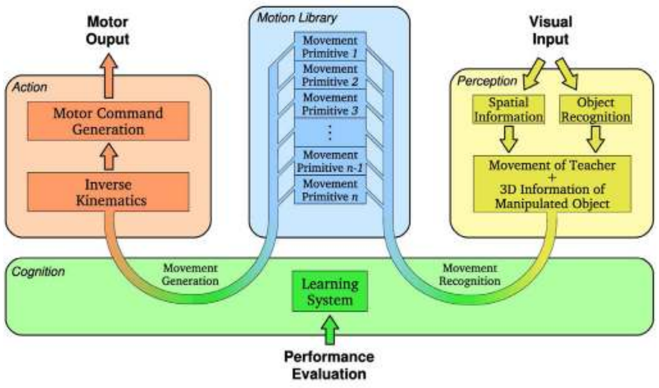

Fig. 3. Conceptual sketch of an imitation learning system (adapted from [1]). The components of perception (yellow) transform visual information into spatial and object information. The components of action (red) generate motor output. Interaction between them is achieved using a common motion library (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.

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 [5]. 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 (Fig. 3). This makes the presented formulation suited for imitation learning.

B. Attaching Semantic

For imitation learning with DMPs, we chose a low-level approach, namely imitation of trajectories [2]. However, 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 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

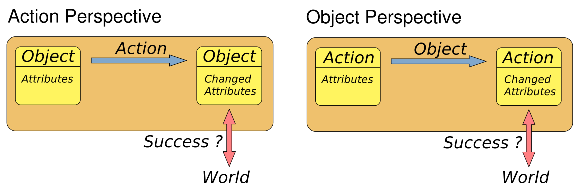

Fig. 4. Objects are defined through actions that can be performed on them (Left), e.g. a cup is represented as a thing which can be used to drink water from. On the other side, actions are defined through objects (Right), e.g. the way of grasping an object depends on the object - a can requires a different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

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 [11], [12]. This concept proposes that objects and actions are inseparably intertwined (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 (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 $\mathbf{x}_{\text{pred}} \rightarrow \mathbf{x}_{\text{succ}}$).

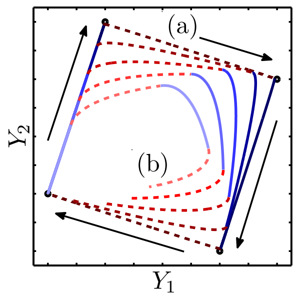

Fig. 5. Chaining of four straight minimum-jerk movement primitives, whose endpoints are marked by black dots (a). The movements generated by the DMPs are drawn alternating blue solid and red dashed lines to indicate the transition between two successive DMPs. The movement direction is indicated by arrows. The remaining movements (b) result from using different switching times (a lighter color indicates an 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 (Fig. 6) equipped 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 movements 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

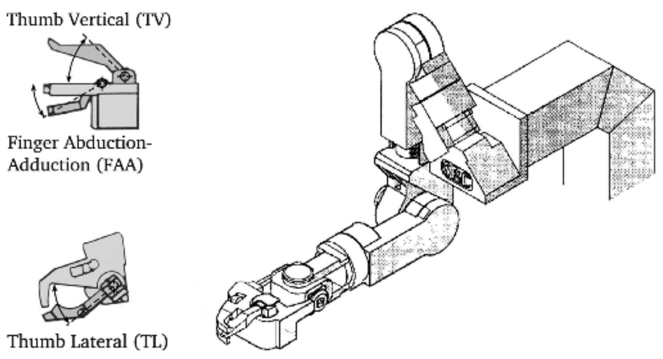

Fig. 6. Sketch of the Sarcos Slave arm, a seven DOF anthropomorphic robot arm with a three DOF end-effector.

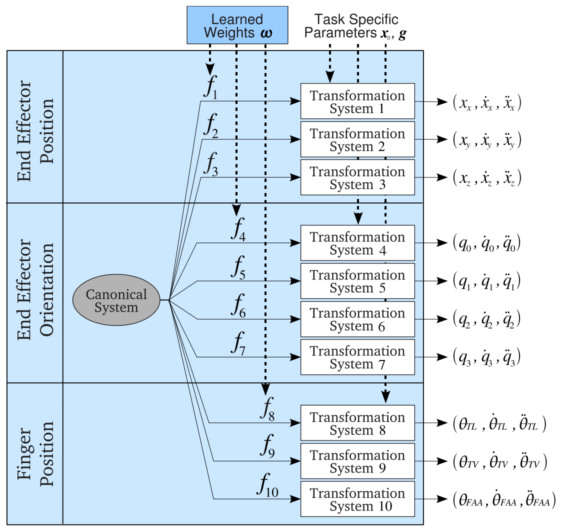
Learned Weights $\boldsymbol{\omega}$
Task Specific Parameters $\boldsymbol{x}_0$, $\boldsymbol{g}$
End Effector Position
End Effector Orientation
Finger Position
Canonical System
f_1
f_2
f_3
f_4
f_5
f_6
f_7
f_8
f_9
f_{10}
Transformation System 1
Transformation System 2
Transformation System 3
Transformation System 4
Transformation System 5
Transformation System 6
Transformation System 7
Transformation System 8
Transformation System 9
Transformation System 10
$(x_x, \dot{x}_x, \ddot{x}_x)$
$(x_y, \dot{x}_y, \ddot{x}_y)$
$(x_z, \dot{x}_z, \ddot{x}_z)$
$(q_0, \dot{q}_0, \ddot{q}_0)$
$(q_1, \dot{q}_1, \ddot{q}_1)$
$(q_2, \dot{q}_2, \ddot{q}_2)$
$(q_3, \dot{q}_3, \ddot{q}_3)$
$(\theta_{TL}, \dot{\theta}_{TL}, \ddot{\theta}_{TL})$
$(\theta_{TV}, \dot{\theta}_{TV}, \ddot{\theta}_{TV})$
$(\theta_{FAA}, \dot{\theta}_{FAA}, \ddot{\theta}_{FAA})$

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

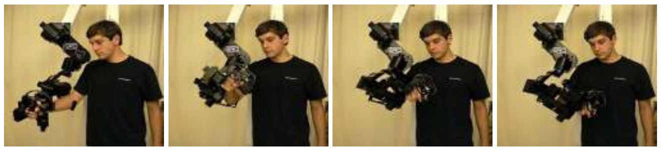

Fig. 8. Sarcos Master arm used to record a human trajectory in end-effector space. Here, the subject demonstrates a pouring movement which after learning the DMP enabled a robot to pour water into several cups (Fig. 12).

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 (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.

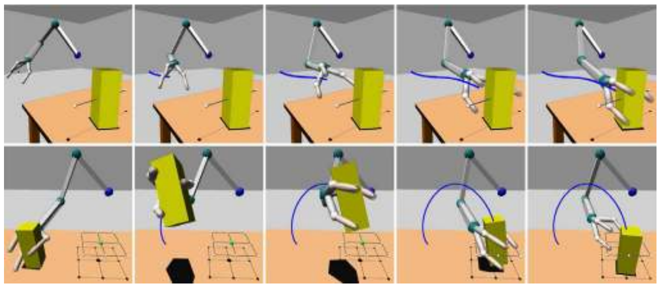

Fig. 9. Snapshots of the *SL* Simulator showing a simulation of the Sarcos Slave arm performing a grasping (Top) and a placing movement (Bottom).

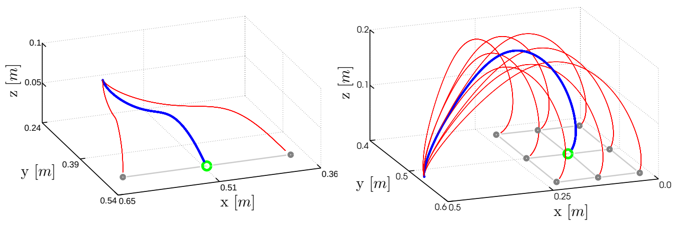

Fig. 10. The desired trajectories (blue lines) from the movements shown in Fig. 9 adapted to new goals (red lines) indicated by the grid.

C. Task Space Control

To execute DMPs on the robot we used a velocity based inverse kinematics controller as described in [13], [8]. Thus, the task space reference velocities $\dot{\boldsymbol{x}}_r$ are transformed into the reference joint space velocities $\dot{\boldsymbol{\theta}}_r$ (Fig. 11). The reference joint position $\boldsymbol{\theta}_r$ and acceleration $\ddot{\boldsymbol{\theta}}_r$ are obtained 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 [14], [8].

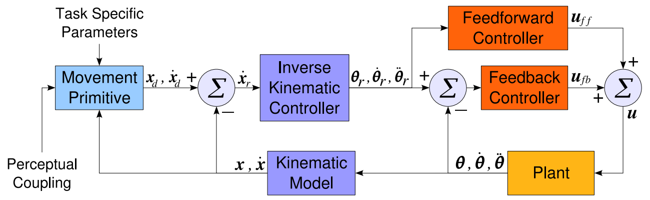
Task Specific Parameters
Movement Primitive
Perceptual Coupling
$\boldsymbol{x}_d, \dot{\boldsymbol{x}}_d$
+
Σ
−
$\dot{\boldsymbol{x}}_r$
Inverse Kinematic Controller
$\boldsymbol{\theta}_r, \dot{\boldsymbol{\theta}}_r, \ddot{\boldsymbol{\theta}}_r$
Feedforward Controller
$\boldsymbol{u}_{ff}$
Feedback Controller
$\boldsymbol{u}_{fb}$
$\boldsymbol{u}$
$\boldsymbol{x}, \dot{\boldsymbol{x}}$
Kinematic Model
$\boldsymbol{\theta}, \dot{\boldsymbol{\theta}}, \ddot{\boldsymbol{\theta}}$
Plant

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 (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 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 (noise: SD = 0.001 m

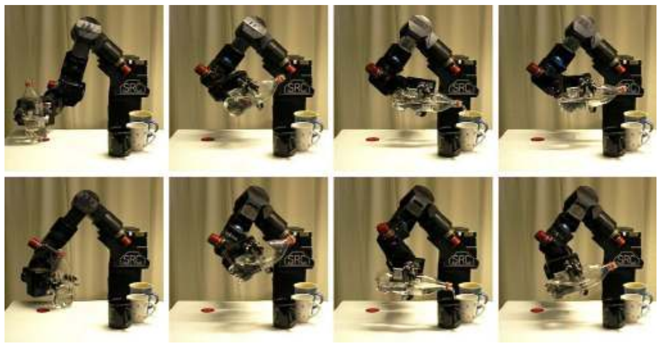

Fig. 12. Movement reproduction and generalization to new goal with the Sarcos Slave Arm. The top row shows the reproduction of a demonstrated pouring movement in Fig. 8, and the bottom row shows the result of changing the goal variable.

per dimension). 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 (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 that changed their position during the robot's movement. Additionally, movement trajectories were automatically adapted to avoid moving obstacles (Fig. 13 and video supplement).

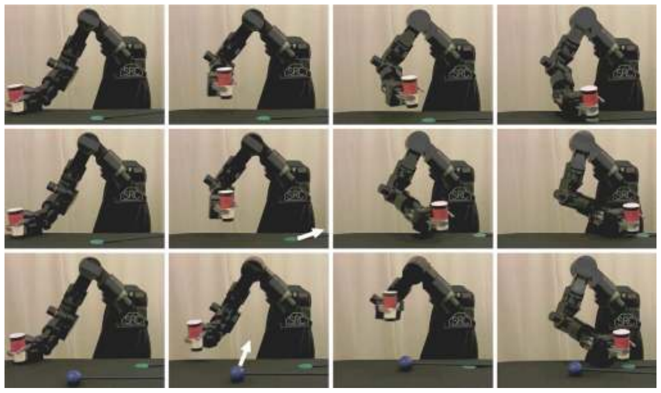

Fig. 13. Sarcos Slave arm placing a red cup on a green coaster. The first row shows the placing movement on a fixed goal. The second row shows the resulting movement as the goal changes (white arrow) after movement onset. The third row shows the resulting movement as a blue ball-like obstacle interferes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

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: robust generalization to new goals, human like adaptation, and automatic obstacle avoidance. Moreover, we 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 [12]) 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, "Special issue on robot learning from demonstration," *Robotics and Autonomous Systems*, vol. 47, pp. 65–67, 2004.

[3] J. Demiris and G. Hayes, "Imitative Learning Mechanisms in Robots and Humans," in *Proceedings of the European Workshop on Learning Robotics*, 1996.

[4] C. Nehaniv and K. Dautenhahn, *Imitation in Animals and Artifacts*. MIT Press, 2002.

[5] 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*, 2002.

[6] 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.

[7] H. Hoffmann, P. Pastor, D. Park, and S. Schaal, "Biologically-inspired dynamical systems for movement generation: Automatic real-time goal adaptation and obstacle avoidance," in *Proceedings of the IEEE International Conference on Robotics and Automation*, Kobe, Japan, 2009.

[8] 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.

[9] D. Park, H. Hoffmann, P. Pastor, and S. Schaal, "Movement reproduction and obstacle avoidance with dynamic movement primitives and potential fields," in *Proceedings of the IEEE-RAS International Conference on Humanoid Robots*, Daejeon, Korea, 2008.

[10] 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.

[11] 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.

[12] 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*, 2006.

[13] 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.

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