Learning to Select and Generalize Striking Movements in Robot Table Tennis

Katharina Muelling[1,2] and **Jens Kober**[1,2] and **Oliver Kroemer**[2] and **Jan Peters**[1,2]
(1) Max Planck Institute for Intelligent Systems
Spemanstr. 38, 72076 Tübingen, Germany

(2) Technische Universität Darmstadt, FG Intelligente Autonome Systeme
Hochschulstr. 10, 64289 Darmstadt, Germany

Abstract

Learning new motor tasks autonomously from interaction with a human being is an important goal for both robotics and machine learning. However, when moving beyond basic skills, most monolithic machine learning approaches fail to scale. In this paper, we take the task of learning table tennis as an example and present a new framework which allows a robot to learn cooperative table tennis from interaction with a human. Therefore, the robot first learns a set of elementary table tennis hitting movements from a human teacher by kinesthetic teach-in, which is compiled into a set of dynamical system motor primitives (DMPs). Subsequently, the system generalizes these movements to a wider range of situations using our mixture of motor primitives (MoMP) approach. The resulting policy enables the robot to select appropriate motor primitives as well as to generalize between them. Finally, the robot plays with a human table tennis partner and learns online to improve its behavior. We show that the resulting setup is capable of playing table tennis using an anthropomorphic robot arm.

1 Introduction

Learning new motor tasks autonomously and adapting motor skills online while interacting with the environment is an important goal in robotics as well as machine learning. In recent years, it has become well accepted that for coping with the complexity involved in motor skill learning for robots, we need to rely on the insight that humans decompose motor tasks into smaller subtasks. These subtasks can be solved using a small number of generalizable movement pattern generators, also called movement primitives (Flash and Hogan 1985; Giszter et al. 2000; Billard et al. 2008). Movement primitives are sequences of motor commands executed in order to accomplish a given motor task. Efficient learning of movement primitives is crucial as the state space can be high dimensional and the number of scenarios that may need to be explored grows exponentially with the number of dimensions and time-steps (Schaal 1999). Here, learning from demonstrations can provide a good starting point for motor skill learning as it allows the efficient acquisition of single movement primitives. Most approaches for robot imitation learning are either based on physical demonstrations of a motion sequence to the robot by kinesthetic teach-in (Guenter et al. 2007; Peters and Schaal 2008; Bitzer, Howard, and Vijayakumar 2010) or through the use of a motion capture system such as a VICON setup (Kober, Mohler, and Peters 2008; Gräve, Stückler, and Behnke 2010). We refer to (Schaal, Ijspeert, and Billard 2003; Billard et al. 2008; Argall et al. 2009) for a review of imitation learning methods.

To represent movement primitives such that they can adapt trajectories obtained by imitation learning, several methods have been suggested (Miyamoto et al. 1996; Ijspeert, Nakanishi, and Schaal 2002; Calinon, Guenter, and Billard 2007; Williams, Toussaint, and Storkey 2008). Among them an approach based on dynamical systems was suggested by Ijspeert, Nakanishi, and Schaal (2002) and called dynamical system motor primitives (DMPs). DMPs are robust against perturbations, allow the change of the final state, speed and duration without the need of changing the overall shape of the movement. Furthermore, they are straightforward to learn by imitation learning (Ijspeert, Nakanishi, and Schaal 2002) and well suited for reward driven self-improvement (Kober and Peters 2009). DMPs have been successfully used to learn a variety of motor skills in robotics, including planar biped walking (Nakanishi et al. 2004), T-ball batting (Peters and Schaal 2006), constrained reaching tasks (Guenter et al. 2007), and Ball-in-a-cup (Kober, Mohler, and Peters 2008). However, up to now, most applications of learning and self-improving Ijspeert's DMPs use only individual movement primitives to represent the whole motor skill. An exception is the work of Ude et al. (2010), in which the internal parameters of the DMPs are recomputed from a library of movement primitives in each trial using locally weighted regression. However, complex motor tasks require several movement primitives which are used in response to an environmental stimulus and the usage needs to be adapted according to the performance.

In this paper, we attempt to create such a framework based on the idea that complex motor tasks can frequently be solved using a relatively small number of movement primitives (Flash and Hogan 1985) and do not require a complex monolithic approach. The goal of the paper is to acquire a library of movement primitives from demonstration (to which we will refer as movement library), and to select and gen-

eralize among these movement primitives to adapt to new situations. Each movement primitive stored in the library is associated with a set of parameters to which we refer as the augmented state that describes the situation present during demonstration. The primitives in the movement library are used as components in our mixture of motor primitives (MoMP) algorithm. The MoMP algorithm activates components (i.e., single movement primitives) using a gating network based on the augmented state and generates a new movement using the activated components. The activation of the components can be updated online based on the performance of the system. Our approach is validated using robot table tennis as a benchmark task. The hitting movements in table tennis may vary depending on the point of impact relative to the base of the robot, the time available until impact or the kind of stroke that should be performed. Furthermore, small inaccuracies in timing can lead to large deviations in the final bouncing point of the returned ball that result in unsuccessful attempts return the ball to the opponent's court. The goal of this task is to learn autonomously from and with a human to return a table tennis ball to the opponent's court and to adapt its movements accordingly. Therefore, the robot first learns a set of striking movements from a human teacher from physical human robot guiding, known as kinesthetic teach-in. From this stream of data, the movement primitives were extracted. Secondly, the learning system identifies the augmented state that includes where, when and how the ball should be hit. Subsequently, the system generalizes these movements to a wider range of situations using the proposed MoMP algorithm. Here, generalizing refers to the ability to generate striking movements towards any given goal on the opponent's court for an arbitrary incoming ball served to the forehand area of the robot. The resulting system is able to return balls served by a ball launcher as well as to play in a match against a human. A video can be found at `www.youtube.com/watch?v=SH3bADiB7uQ`.

In the remainder of the paper, we will proceed as follows. In Section 2, we present our general framework for learning complex motor skills using the MoMP algorithm. We evaluate the MoMP approach in a robot table tennis scenario in Section 3. In Section 4, we will present our results and summarize our approach.

2 Learning Motor Behaviors

In a complex motor task such as table tennis, we need to coordinate different movements which highly depend on a changing context. Unlike in many classical examples (Peters and Schaal 2008; Pongas, Billard, and Schaal 2005; Nakanishi et al. 2004), single movement primitives which were demonstrated to work well in a certain situation do not necessarily perform equally well in other situations that might occur throughout the task. In table tennis, the movement profile depends strongly on where, when and how the ball has to be hit, as well as the velocity of the incoming ball and the desired target on the opponent's court. As it is not feasible to demonstrate all possibly required movements to the robot, the system needs to generalize from a much smaller number of movement primitives. Hence, we suggest an algorithm called mixture of motor primitives (MoMP)

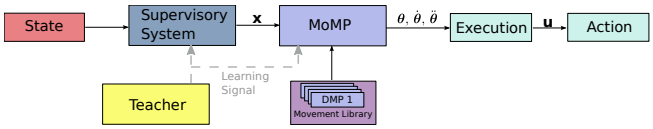


Figure 1: General setup for learning a motor task using the MoMP framework. A supervisory level creates the augmented state $\mathbf{x}$ containing the relevant information of the task based on the state of the system. The MoMP algorithm selects and generalizes among the movement templates in a library based on $\mathbf{x}$. As a result we obtain a new motor policy that can be executed. A teacher provides learning signals to the supervisory level as well as the movement generation level.

which generates a generalized movement for a given augmented state that can be executed by a robot. Therefore, a set of movement primitives and their corresponding augmented state are extracted from a set of demonstrations. Both the movement primitives and their corresponding augmented states are stored in a library.

To generate a movement for a new augmented state $\mathbf{x}$ (that was not presented during demonstration), the system selects movement primitives from the library. Therefore, a parametrized gating network is used in the MoMP algorithm to activate movement primitives based on the presented augmented state. In some cases, the augmented state might be directly available as part of the state of the system. However, in table tennis, additionally parameters $\boldsymbol{\delta}$ need to be estimated from the state by a supervisory system. The state of the system $\mathbf{s}$ consists of all variables necessary to model the system, e.g., in our table tennis task, the position and velocity of the ball moving towards the robot and the current joint configuration of the robot itself. The additional parameters $\boldsymbol{\delta}$ of the augmented state $\mathbf{x}$ are given by the point of impact, the velocity and orientation of the racket at the hitting point and the time until hitting the ball.

Both, the supervisory system which generates the augmented state and the gating network of the MoMP algorithm that selects and mixes the motor primitives according to the augmented state may need to be adapted to improve the performance of the system. Figure 1 illustrates the general setup for executing a motor task based on the current state of the system and the relation between the state, the augmented state and the movement generation.

In the remainder of this section, we will first present the MoMP framework (Section 2.1). Subsequently, we explain how to compute the augmented state (Section 2.2). Finally, we show how to use and initialize DMPs as elementary motor policies in the MoMP framework (Section 2.3).

2.1 Learning a Task with a Mixture of Motor Primitives

A movement performed by an artificial or biological system can be formalized as a policy

$$\mathbf{u} = \pi(\mathbf{x}, \mathbf{w})$$

that maps the state of the system, described by vector $\mathbf{x} =$

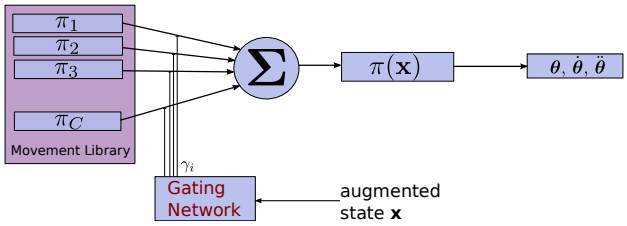


Figure 2: The mixture of motor primitive framework. The gating network weights the single movement templates stored in a movement library based on an augmented state. The weighted sum of these primitives defines the new motor policy which produces the joint positions, velocities and accelerations for one degree of freedom.

$[\mathbf{s}, \boldsymbol{\delta}]$, to a control vector $\mathbf{u}$. The vector $\mathbf{w}$ contains task specific adjustable parameters. In the following, we will refer to such a policy as motor policy $\pi(\mathbf{x})$. The state of the system $\mathbf{x}$ corresponds here to the augmented state.

We generate the motor policy $\pi(\mathbf{x})$ based on a library consisting of L movement primitives. Each movement primitive $i \in \{1, ..., L\}$ in the library is stored in a motor policy π_i. Additionally, for each motor policy π_i, the augmented state $\mathbf{x}_i$ associated with this movement primitive is stored. The movement library can be initialized using movements demonstrated in different situations of the task.

The MoMP generates a new movement for the current situation triggered by the augmented state $\mathbf{x}$ by computing the weighted average of all movement primitives π_i (see Figure 2). The resulting motor policy generated by MoMP algorithm is given by

$$\pi(\mathbf{x}) = \frac{\sum_{i=1}^{L} \gamma_i(\boldsymbol{\delta}) \pi_i(\mathbf{x})}{\sum_{j=1}^{L} \gamma_j(\boldsymbol{\delta})}, \quad (1)$$

where the function $\gamma_i(\boldsymbol{\delta})$ generates the weight of $\pi_i(\mathbf{x}) = [\mathbf{s}, \boldsymbol{\delta}]$ given the augmented state $\mathbf{x}$. All weights γ_i together form the "gating network" of the MoMP similar to a gating network in a mixture of experts (Jordan and Jacobs 1994).

The weights of the gating network ensure that only movement primitives that are well-suited for the current situation can contribute to the resulting behavior. It appears reasonable to assume that movement primitives with an associated augmented state similar to the currently observed one are more likely to produce successful movements than movement primitives whose augmented states differ significantly from the observation. However, any large set of demonstrations will include rather poor attempts and, thus, some demonstrations are better suited for generalization than others. Therefore, the gating network has to weight the movement primitives based on their expected performance within the current context. Such weights can be modeled by an exponential family distribution

$$\gamma_i(\mathbf{x}) = \xi \exp\{\boldsymbol{\vartheta}_i^T \phi_i(\mathbf{x})\}, \quad (2)$$

where $\phi_i(\mathbf{x})$ denotes the feature vector of $\mathbf{x}$, $\boldsymbol{\vartheta}_i$ is a vector containing internal parameters, and ξ is a normalization constant. The weight γ_i corresponds to the probability that the motor policy π_i is the right policy in the context described by $\mathbf{x}$, i.e., $\gamma_i(\mathbf{x}) = p(\pi_i|\mathbf{x})$. In an ideal world, there would be just one motor policy in the library that is perfectly suited for the current context. However, in practice, usually several motor policies correspond to this context imperfectly. Therefore, the MoMP needs to generalize among these motor policies, i.e., it mixes these movement primitives according to their weights γ_i in order to yield a motor policy that can be used in a broader context.

The choice of $\phi_i(\mathbf{x})$ depends on the task. In our experiments however, a Gaussian basis function where the center is given by the augmented state $\mathbf{x}_i$ proved to be a good choice. The parameters $\boldsymbol{\vartheta}_i$ of the probability distribution $\gamma_i(\mathbf{x})$ are unknown and have to be determined. If good features $\phi(\mathbf{x})$ are known, linear regression methods are well suited to learn the parameters $\boldsymbol{\vartheta}_i$ given examples of γ_i and the corresponding $\phi(\mathbf{x})$. Hence, we use linear Bayesian regression (Bishop 2006) to update the mean and variance of the parameters of the distribution online for each motor policy in the library. The parameters are updated during the execution of the task based on the performance of the system. The performance can be measured by the reward r which is provided by a teacher to the system and corresponds in table tennis to the distance of the returned ball to the desired goal on the opponent's court. As a result, the system is able to adapt the choice of the used motor policies.

Altogether, the mixture of motor primitives selects and generalizes between movement primitives in the library based on the current augmented state $\mathbf{x}$. The resulting motor policy $\pi(\mathbf{x})$ is composed of several primitives weighted by their suitability in the given context of the task. The weights are determined by a gating network and adapted to the task based on the outcome of previous trials.

2.2 Computing the Augmented State

Some parameters required for the task are not part of the state and need to be computed. In table tennis, these parameters include the temporal and spacial interception point of the ball and the racket, as well as the velocity and orientation of the racket while hitting the ball. When planning in joint space, this corresponds to finding the position and velocity at the interception point for each joint. These parameters are an essential part of the generation of a desired movement with the motor policy $\pi(\mathbf{x})$ as they define the final state and duration of the movement. We refer to these additional parameters as the meta-parameter $\boldsymbol{\delta}$. The augmented state $\mathbf{x}$ is given by $\mathbf{x} = [\mathbf{s}, \boldsymbol{\delta}]$.

One possibility of computing the meta-parameters is to predict the trajectory of the ball using an extended Kalman predictor starting with the current state of the system. Subsequently, we can determine a well suited hitting point on the trajectory and compute the desired velocity and orientation of the racket given a target on the opponent's court. Using inverse kinematics is one way to compute the corresponding joint configuration (see Muelling, Kober, and Peters (2011) for a detailed description).

An alternative possibility is to use an episodic reinforcement learning approach to acquire the mapping from $\mathbf{s}$ to the meta-parameter $\boldsymbol{\delta}$ directly. Here, Cost-regularized Ker-

nel Regression (CrKR), see Kober et al.(2012), has also proven to be suited for learning meta-parameters for motor policies as used in this project.

2.3 Behavior Representation with Motor Primitives

To represent a single motor policy π_i used in the MoMP, we employ dynamical system motor primitives (DMPs). DMPs, as suggested in (Ijspeert, Nakanishi, and Schaal 2002; Schaal, Mohajerian, and Ijspeert 2007), are a particular kind of dynamical systems that is well-suited for imitation and reinforcement learning. It can be understood as a set of two differential equations that are referred to as the *canonical* and the *transformed* system. The canonical system h acts as a phase z of the movement generated by

$$\dot{z} = h(z). \tag{3}$$

Intuitively, one could state that the canonical systems drives the transformed system similar to a clock. The transformed system

$$\mathbf{u} = \pi(\mathbf{x}) = d(y, g_f, z, \mathbf{w}), \tag{4}$$

generates the desired movement for a single degree of freedom (DoF). It is a function of the current position y of the system, the final goal position g_f, the phase variable z and the internal parameter vector $\mathbf{w}$. The movement can be specified in joint or task space.

DMPs allow us to represent arbitrarily shaped smooth movements by the parameter vector $\mathbf{w}$, which can be learned from demonstration by locally weighted regression (for a detailed description see Schaal et al. (2003)). Furthermore, it is straightforward to adapt the DMPs with respect to the final position, movement amplitude and duration of the movement without changing the overall shape. As a consequence, we can adapt the movement during execution to new movement goals and time constraints without re-planing the whole movement. However, the original formulation cannot be used for striking movements in a straightforward manner as the formulation does not account for non-zero end-velocities or via-points without changing the shape of the movement. Hence, we need motor primitives that allow for different movement stages where the stages are switched based on features of the state. To achieve this goal, we introduce a type of two-stage motor primitive suited for hitting movements and use the feature of ball-racket contact to allow the system to switch the stage. While Kober et al. (2010) augmented Ijspeert's approach to make it possible to strike moving objects with an arbitrary velocity at the hitting point, this formulatin has two drawbacks. First, movement changes of the final position and velocity can lead to inaccuracies in the final velocity (see Figure 3). Second, if the start and final position in the demonstratin are close to each other, changing the goal position can cause huge accelerations at the beginning of the movement (see Figure 4).

Modified Motor Primitives for Striking Movements For discrete movements (i.e., movements between fixed start and end positions such as reaching, pointing, grasping and striking movements) the canonical system is defined by

$$\tau \dot{z} = -\alpha_z z, \tag{5}$$

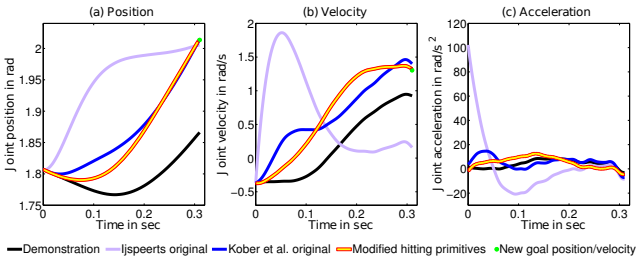


Figure 3: This figure illustrates how the different versions of the dynamical system based movement primitives are modulated by changing the goal position and velocity of the movement. The demonstration of a striking movement was obtained by kinesthetic teach-in in table tennis. All movement primitive formulations were able to reproduce the demonstration. We changed the position by 0.15 m and the velocity by 0.4 m/s. The original formulation of Ijspeert, Nakanishi, and Schaal (2002) is not able to reach the desired velocity. The formulation of Kober et al. (2010) is able to adapt to a new final velocity. However, the accuracy of the adapted movement does not suffice for practical problems. The reformulation presented in this paper reduces this inaccuracy drastically and stays closer to the desired movement shape.

where τ is a temporal scaling factor and α_z is a pre-defined constant which is chosen such that the behavior is stable. Initially z is set to 1 and converges to zero at the end of the movement.

For hitting movements, we propose the transformed system

$$\begin{aligned}
\tau \dot{v} &= \alpha_y \left(\beta_y (g - y) + \dot{g}\tau - v\right) + \ddot{g}\tau^2 + \eta f(z), \\
\tau \dot{y} &= v, \qquad\qquad \eta = \frac{\exp\left(g_f - y_0\right)}{\exp(a)}, \\
g &= \sum_{i=0}^{5} b_i \left(-\tau \frac{\ln(z)}{\alpha_z}\right)^i, \quad \dot{g} = \sum_{i=1}^{5} i b_i \left(-\tau \frac{\ln(z)}{\alpha_z}\right)^{i-1}, \\
\ddot{g} &= \sum_{i=2}^{5} (i^2 - i) b_i \left(-\tau \frac{\ln(z)}{\alpha_z}\right)^{i-2},
\end{aligned} \tag{6}$$

where y and v are the desired position and velocity generated by the policy, η defines the amplitude of the movement, y_0 is the start position, g_f and $\dot{g}_f$ are the desired final position and final velocity of the system, g, $\dot{g}$ and $\ddot{g}$ are the current position, velocity and acceleration defined by the moving target, $\tau \ln(z)/\alpha_z$ corresponds to the time and a is a reference amplitude. If the internal parameters $\mathbf{w}$ are estimated by imitation learning as in the experiments performed in this paper, a will correspond to the amplitude of the demonstrated movement. The parameters b_j are computed by applying the bounding conditions, i.e., the condition that the fith order polynomial starts with the initial position, velocity and acceleration and ends at the desired goal g_f with the desired velocity $\dot{g}_f$ and zero acceleration. The pre-defined spring-damper constants α_y and β_y are chosen such that the system

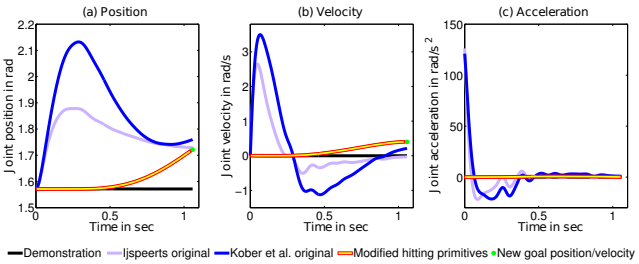


Figure 4: A key problem of the previous movement primitive formulation is the highly uneven distributed accelerations with a peak at the beginning of the movement. Such jumps can affect the position and velocity drastically as shown in this figure. They may result in the attempt to generate infeasible trajectories for real robots. Kober et al. reduced this effect by gradually activating the attractor dynamics. However, if the initial movement amplitude is close to zero in a demonstration jumps will occur when the goal position is changed.

is critically damped. The transformation function

$$f = \frac{\sum_{j=1}^{N} w_j \psi_j(z) z}{\sum_{j=1}^{N} \psi_j(z)}, \tag{7}$$

employs Gaussian basis functions $\psi_j(z) = \exp(-\rho_j(z - \mu_j)^2)$ characterized by a center μ_j and a bandwidth ρ_j. N is the number of adjustable parameters w_j. The transformation function f alters the output of the spring damper model and thereby allows the generation of arbitrarily shaped movements. As z converges to zero at the end of the movement, the influence of the non-linear function f will vanish. The augmented form of the DMP enables us to pull the DoF simultaneously to a desired goal position and an arbitrary end velocity without changing the overall shape of the movement or its duration. Using a fifth order polynomial allows us to control the initial and final position, velocity and acceleration. The scaling term $\eta = \exp(g_f - y_0)/\exp(a)$ ensures that f does not cause infeasible acceleration. In the following we will refer to the acceleration yielded by the DMP i as motor policy π_i.

3 Evaluation

In Section 2, we have described a framework for selecting and generalizing movements based on augmented states as well as for computing the meta-parameters which are part of the augmented states from the state information of the environment. Here, we will show that we can use these methods to learn robot table tennis from and with a human being. Therefore, we will first give a short overview of the table tennis task and then evaluate the methods in simulation as well as on a real robot.

3.1 Robot Table Tennis Setup

For the robot table tennis task, we developed a system that includes a Barrett WAM arm with seven DoFs capable of high speed motion for hitting the ball and a vision system with four Prosilica Gigabit GE640C cameras for tracking the ball (Lampert and Peters 2012). The robot is mounted in a hanging position from the ceiling. A standard racket, with a 16 cm diameter, is attached to the end-effector. The setup incorporates a standard sized table tennis table and a table tennis ball in accordance with the International Table Tennis Federation (ITTF) rules (International Table Tennis Federation 2011). The ball is served either by a ball launcher to the forehand of the robot with a randomly chosen velocity or served by a human. The area covered is approximately 1 m^2. The ball is visually tracked with a sampling rate of 60 Hz and the vision information is filtered using an extended Kalman filter (EKF). For the internal model of the EKF, we assume a simplified model of the flight and bouncing behavior of the ball, i.e., we consider gravity and air drag, but neglect the spin acting on the ball due to its limited observability. The world frame is fixed at the base of the robot, with the negative y-axis pointing towards the opponent and the z-axis pointing upwards.

If the visual system detects a table tennis ball that moves towards the robot, the system needs to compute the relevant movement information as where, when and how the ball has to be hit (see Section 2.2). The target location on the opponent's court was fixed to the center of the court to make the results comparable.

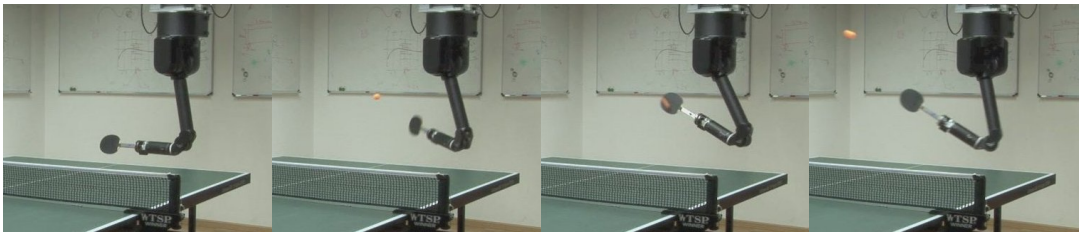
3.2 Computing the Meta-Parameters

To use DMPs as motor policies in a table tennis game, the system needs to identify the hitting position, velocity and orientation of the racket, as well as the time until impact. These parameters need to be estimated for each incoming ball in a match. When planning in joint space, the hitting position, velocity and orientation of the racket are defined by the joint configuration of the robot at this time point. Altogether, 15 parameters need to be determined, which include the timing parameter that determines the initiation of the hiting movement. These values depend on the impact point of the racket and the ball, the velocity of the ball and the desired goal on the court of the opponent. As the goal on the opponent's court is kept constant, we can neglect this parameter. A detailed description and evaluation of learning the parameters using CrKR for this table tennis setup can be found in Kober et al. (2012). Besides learning the meta-parameters directly, the position, velocity and orientation of the racket can be computed analytically based on the state of the system and the target on the opponent's court. These task space parameters can also be converted into joint space parameters using inverse kinematics (Muelling, Kober, and Peters 2011).

3.3 MoMP

To prove the performance of our system described in Section 2, we analyzed the MoMP framework in simulation as well as on a real Barrett WAM in a table tennis setup. The system learns a set of basic hitting movements from demonstration and, subsequently, generalizes these demonstrations to a wider range of situations.

(a) Physical human robot interaction: kinesthetic teach-in of a striking motion in table tennis.



(b) Reproduced hitting motion by imitation learning.

Figure 5: Sequence of a hitting motion in table tennis demonstrated by a human teacher and reproduced with a Barrett WAM arm with seven DoF. From the left to the right the single pictures represent the system at the end of the awaiting, preparing, hitting and follow through stage respectively.

Evaluation in Simulation. To evaluate our setup as described in Section 2 in a variety of different situations under near-perfect conditions, we first evaluate it in simulation. The parameters of the hitting movement depend on the interception point of the racket and the ball and vary in their overall shape. To generate a movement that is able to cope with the varying conditions, we use our MoMP framework with the estimated hitting point and velocities as augmented state. The movement library was built using 300 movement templates sampled from successful strokes of an analytical robot table tennis player (Muelling, Kober, and Peters 2011). For the simulation of the ball, we neglected air drag in this evaluation. The system had full knowledge of the correct position and velocity of the ball. Note that this reduces the number of potential error sources. Thus, the success rate of returning the ball back to the desired goal on the opponent's court reflects the performance of the algorithm more accurately. We collected arm, racket and ball trajectories and extracted the duration of the submovements and the Cartesian ball positions and velocities at the hitting point. The parameters $\mathbf{w}$ for all movement primitives were learned offline by imitation learning. All DoFs were modeled independently in the transformed system but are synchronized such that they all start at the same time, have the same duration and are driven by the same canonical system. The Cartesian position and velocity of the expected hitting point were used as meta-parameters of the augmented state. The balls were served equally distributed on an area of 1.1 m x 0.3 m.

First, we evaluated the single mixture components (i.e., the movement primitives learned by imitation learning) independently. Testing randomly selected movement primitives individually, we observed a success rate of 23 % to 89 %, where success was defined as the ability to return the ball to the opponent's court. The combination of these components in an untrained MoMP algorithm resulted into a player which achieved a success rate of 67 %. Learning of the weight parameters γ_i over 1000 trials improved the performance to 94 %.

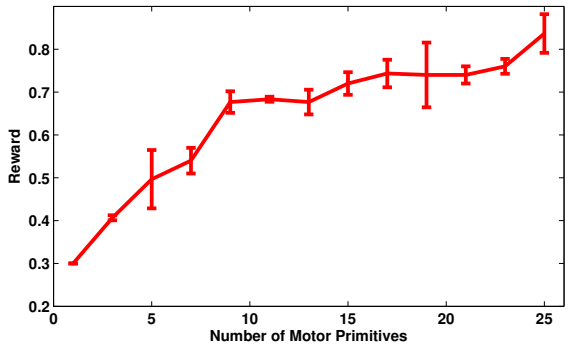


Figure 6: Improvement of the MoMP system with a growing number of movement primitives in the library.

Real Robot Table Tennis. We evaluated our setup on a Barrett WAM arm. Kinesthetic teach-in was used to record 25 striking motions which all started at the same initial position (see Figure 5). Gravity compensation of the robot was enabled, but no additional movement commands were sent to the robot. As a result, the robot could be moved freely by the teacher. For the demonstrations, the ball was served by a ball launcher covering the forehand area of the robot. The recorded arm movements were divided into submovements according to the following events: contact with the ball, zero velocity and change in movement direction. As a result, each demonstration could be divided into four submovements:

preparing the hitting by moving the racket backwards, hitting the ball in a circular forward movement, follow throw until the velocity goes to zero and moving back to the default position. We created the movement library of DMPs for each movement recorded from the human teacher using imitation learning. As augmented state, we used the estimated hitting position and velocity.

We evaluated the performance of the created movement library intensively in simulation first. Here, we used the exponential of the negative distance between the desired joint velocity and the actual joint velocity of the last three DoFs[1] at the interception point of the ball and the racket. The balls were served to an area of 1.1 m x 0.3 m. The system was trained using 100 trials and evaluated over 200 additional trials. The average performance of the system using MoMP without updating the gating network was 0.08 rad/s. Updating but just choosing the movement primitive with the best expected performance, we had an error of 0.05 rad/s and mixing the movement primitives yielded an error of 0.02 rad/s. An overview of the improvement of the system with increasing number of movement primitives used is shown in Figure 6.

Performing the movement on a real system, we used the exponential of the negative distance between the desired goal on the opponent's court and the actual bouncing point as reward signal. If the ball missed the table, the distance would be set to 5 m and the reward was set close to zero. We evaluated the algorithm in two experiments. In the first experiment we used the ball launcher to serve the ball to the robot to ensure similar conditions for both initial and final test phases. We chose to fix the ball launcher in a position where the system was not able to return the ball using the initial policy generated by MoMP. The area served by the ball launcher was 0.25 m x 0.25 m. Initially the system was not able to return any of these balls. After training during 60 trials, the system was able to return 79% of the balls successfully to the opponent's court. The mean distance between the bouncing point of the returned balls and the desired target on the table was 0.31 m. During the training, the usage of the applied movement primitives changed drastically. While some of the movement primitives were relocated, other movement primitives were avoided completely and replaced by others used instead (see Figure 7).

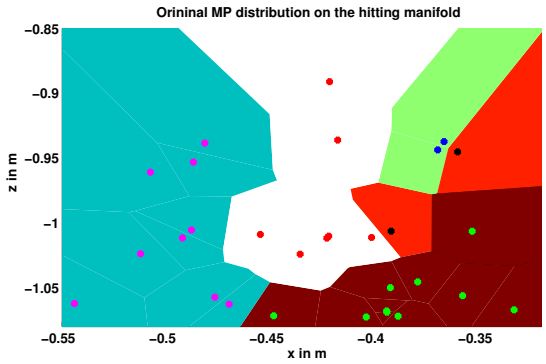
In a third experiment, a human played against the robot. The human served balls on an area of 0.8 m x 0.6 m. The robot hit back up to 9 balls in a row in a match against the robot. Initially the robot was able to return 74.4 % of the balls. After playing one hour against the human, the robot was able to return 88 % of the balls successfully, i.e., to the opponent's court.

4 Conclusion

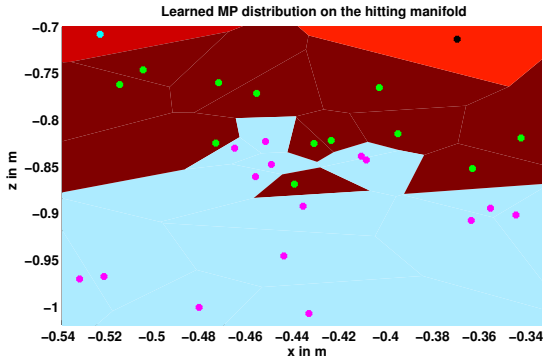
In this paper, we presented a new framework that enables a robot to learn basic cooperative table tennis. To achieve this goal, we created an initial movement library from kinesthetic teach-in and imitation learning. The movements stored in the movement library can be selected and generalized using a mixture of movement primitives algorithm. As a result, we obtain a task policy that is composed of several movement primitives weighted by their ability to generate successful movements in the given task context. These weights are computed by a gating network and can be updated autonomously.

The setup was evaluated successfully in a simulated and real table tennis environment. We showed in our experiments that both (i) selecting movement primitives from a library based on the current task context instead of using only a single demonstration and (ii) the adaptation of the selection process during a table tennis game improved the performance of the table tennis player. As a result, the system was able to perform a game of table tennis against a human opponent.

[1]Please note, that we choose only the last three DoFs, as the peroformance of these DoF were the most critical in this application.



(a)



(b)

Figure 7: Distribution on the hitting manifold of the used movement primitives before (a) and after (b) training. Each color corresponds to one movement primitive in the movement library. Each point corresponds to the point of impact during evaluation.

References

Argall, B.; Chernova, S.; Veloso, M.; and Browning, B. 2009. A survey of robot learning from demonstration. *Robotics and Autonomous System* 57(5):469 – 483.

Billard, A.; Calinon, S.; Dillmann, R.; and Schaal, S. 2008. *Robot Programming by Demonstration*. Springer. 1371–1394.

Bishop, C. 2006. *Pattern Recognition and Machine Learning*. Springer.

Bitzer, S.; Howard, M.; and Vijayakumar, S. 2010. Using dimensionality reduction to exploit constraints in reinforcement learning. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*.

Calinon, S.; Guenter, F.; and Billard, A. 2007. On learning, representing, and generalizing a task in a humanoid robot. *IEEE Transactions on Systems, Man and Cybernetics* 32(2):286–298.

Flash, T., and Hogan, N. 1985. The coordination of arm movements: an experimentally confirmed mathematical model. *Journal of Neurosciences* 5(7):1688–1703.

Giszter, S.; Moxon, K.; Rybak, I.; and J., C. 2000. Neurobiological and neurorobotic approaches to control architectures for a humanoid motor system. *Intelligent Systems and their Applications* 15:64 – 69.

Gräve, K.; Stückler, J.; and Behnke, S. 2010. Learning motion skills from expert demonstrations and own experience using gaussian process regression. In *Joint International Symposium on Robotics (ISR) and German Conference on Robotics (ROBOTIK)*.

Guenter, F.; Hersch, M.; Calinon, S.; and Billard, A. 2007. Reinforcement learning for imitating constrained reaching movements. *Advanced Robotics, Special Issue on Imitative Robots* 21(13).

Ijspeert, A. J.; Nakanishi, J.; and Schaal, S. 2002. Movement imitation with nonlinear dynamical systems in humanoid robots. In *Proc. Int. Conf. on Robotics and Automation (ICRA)*.

International Table Tennis Federation. 2011. Table tennis rules.

Jordan, M., and Jacobs, R. 1994. Hierarchical mixures of experts and the em algorithm. *Neural Computation* 6:181 – 214.

Kober, J., and Peters, J. 2009. Policy search for motor primitives in robotics. In *Advances in Neural Information Processing Systems 21*.

Kober, J.; Muelling, K.; Kroemer, O.; Lampert, C.; Schölkopf, B.; and Peters, J. 2010. Movement templates for learning of hitting and batting. In *Proc. Int. Conf. on Robotics and Automation (ICRA)*.

Kober, J.; Wilhelm, A.; Oztop, E.; and Peters, J. 2012. Reinforcement learning to adjust parametrized motor primitives to new situations. *Autonomous Robots* Epub ahead.

Kober, J.; Mohler, B.; and Peters, J. 2008. Learning perceptual coupling for motor primitives. In *Proc. Int. Conf. on Intelligent Robots and Systems (IROS)*.

Lampert, C., and Peters, J. 2012. Real-time detection of colored objects in multiple camera streams with off-the-shelf hardware components. *Journal of Real-Time Image Processing* 7(1):31–41.

Miyamoto, H.; Schaal, S.; Gadolfo, F.; Gomi, H.; Koike, Y.; Osu, R.; Nakano, E.; Wada, Y.; and Kawato, M. 1996. A kendama learning robot based on bi-directional theory. *Neuronal Networks* 9:1281–1302.

Muelling, K.; Kober, J.; and Peters, J. 2011. A biomimetic approach to robot table tennis. *Adaptive Behavior* 19(5):359 – 376.

Nakanishi, J.; Morimoto, J.; Endo, G.; Cheng, G.; Schaal, S.; and Kawato, M. 2004. Learning from demonstration and adaption of biped locomotion. *Robotics and Autonomous Systems (RAS)* 47(2-3):79 – 91.

Peters, J., and Schaal, S. 2006. Policy gradient methods for robotics. In *Proc. Int. Conf. on Intelligent Robots and Systems (IROS)*.

Peters, J., and Schaal, S. 2008. Natural actor-critic. *Neurocomputing* 71(7-9):1180–1190.

Pongas, D.; Billard, A.; and Schaal, S. 2005. Rapid synchronization and accurate phase-locking of rhythmic motor primitives. In *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, 2911–2916.

Schaal, S.; Peters, J.; Nakanishi, J.; and Ijspeert, A. 2003. Learning motor primitives. In *International Symposium on Robotics Research*.

Schaal, S.; Ijspeert, A.; and Billard, A. 2003. Computational approaches to motor learning by imitation. *Philosophical Transaction of the Royal Society of London: Series B, Biological Sciences* 358:537 – 547.

Schaal, S.; Mohajerian, P.; and Ijspeert, A. 2007. Dynamics systems vs. optimal control – a unifying view. *Progress in Brain Research* 165(1):425 – 445.

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

Ude, A.; Gams, A.; Asfour, T.; and Morimoto, J. 2010. Task-specific generalization of discrete and periodic dynamic movement primitives. *IEEE Transactions on Robotics* 26(5):800 – 815.

Williams, B.; Toussaint, M.; and Storkey, A. 2008. Modelling motion primitives and their timing in biologically executed movements. In *Advances in Neural Information Processing Systems 20 (NIPS)*, 1609–1616.