10th IEEE-RAS International Conference on Humanoid Robots (HUMANOIDS 2010)

Learning Table Tennis with a Mixture of Motor Primitives

Katharina Muelling, Jens Kober, Jan Peters

Abstract— **Table tennis is a sufficiently complex motor task for studying complete skill learning systems. It consists of several elementary motions and requires fast movements, accurate control, and online adaptation. To represent the elementary movements needed for robot table tennis, we rely on dynamic systems motor primitives (DMP). While such DMPs have been successfully used for learning a variety of simple motor tasks, they only represent single elementary actions. In order to select and generalize among different striking movements, we present a new approach, called Mixture of Motor Primitives that uses a gating network to activate appropriate motor primitives. The resulting policy enables us to select among the appropriate motor primitives as well as to generalize between them. In order to obtain a fully learned robot table tennis setup, we also address the problem of predicting the necessary context information, i.e., the hitting point in time and space where we want to hit the ball. We show that the resulting setup was capable of playing rudimentary table tennis using an anthropomorphic robot arm.**

I. INTRODUCTION

While humans are able to perform a variety of complex motor tasks with changing environmental conditions, current robots mostly rely strongly on well-modeled environments in order to perform simple motor skills. In order to cope with the complexity involved in motor skill learning for humanoid robots, we rely on the insight that humans use a smaller number of generalizable movement patterns, also called motor primitives.

Motor primitives that are based on dynamic systems have been suggested by Ijspeert [1]. They have been successfully used in robotics in a variety of different application, including planar biped walking [2] [3], tennis-like swings to a static end-point [1], T-ball batting [4], constrained reaching tasks [5], and Ball-in-a-cup [6]. Nevertheless, most work on motor primitive learning to date has focused on learning single motor primitives. Complex motor tasks, such as table tennis, require many different motor primitives for accomplishing the task. Usually, we have to select the appropriate motor primitive based on an environmental stimulus and, to generalize between several motor primitives to synthesize a new appropriate movement.

Here, we will present an approach for learning complex motor tasks that allows us to select and generalize among motor primitives. This approach relies on a library of motor primitives which are used as components in a Mixture of Motor Primitives (MoMP). The MoMP is activated by a gating network based on external stimuli associated with a single motor primitive. The general setup of our motor learning system is illustrated in Figure 1. To validate our approach we use table tennis as a benchmark task as it requires all complex parts needed. The goal is to create a robot skill learning system that is able to return a ball served by a table tennis ball launcher to the opponent's courts. Therefore, we will also address the question of predicting the necessary context parameters of a motor primitive such as the interception point and time as well as the corresponding joint angles and velocities of the robot at the hitting point. Another work dealing with motion generation using a library of striking motions is the work of Zordan and Hodgins [7]. They simulated human table tennis using motion capture data from table tennis and a balance controller. For frameworks dealing with ball juggling and hitting an object the reader may be interested in the work of Aboaf et al. [8], Kober et al. [9], Frese et al. [10], Andersson [11] and Fässler [12].

In this paper, we will proceed as follows. In Section II, we will present the Mixture of Motor Primitives framework. Therefore, we first introduce the general idea of dynamic systems motor primitives. Subsequently, we described the learning methods needed to learn such a MoMP. The methods used for learning the mapping from context information to MoMP meta-parameters is described in Section II-C. We evaluate the MoMP approach in a robot table tennis scenario in Section III. Here, we will use all components of the presented motor skill learning framework to achieve this task. In Section IV, we will summarize our approach as well as our results and give an outlook on the next steps ahead.

II. MIXTURE OF MOTOR PRIMITIVES (MOMP)

Up to now most work on the application of Ispeert's dynamic systems motor primitives has employed single motor primitives with the notable exceptions of [13], [14]. However, complex motor tasks require several motor primitives that are used in response to certain environmental stimuli. For example, in table tennis many different strikes exist that return the ball to the opponent's court as we can play a smash, a flip, a loop drive, a chop, a block, etc.

As we cannot demonstrate every single trajectory the robot is supposed to execute, we need to generalize between a smaller number of motor primitives. We employ several motor primitives using a central supervisory process that selects the motor primitive and its parameters according to the current context information (Figure 1). To select and generalize the motor primitives, we have developed the MoMP which was inspired by the mixture of experts [15] approach. Skill representation in MoMP is presented in Section II-A and the corresponding learning algorithms are discussed in Section II-B.

All authors are with the Department of Empirical Inference, Max Planck Institute for Biological Cybernetics, Spemannstr. 38, 72076 Tübingen, Germany, email: firstname.lastname@tuebingen.mpg.de

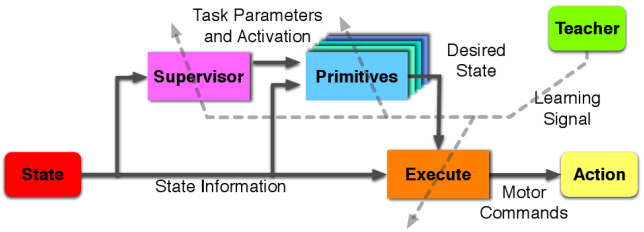


Fig. 1. The Mixture of Motor Primitives motor learning setup. The supervisor level creates a mixture of motor primitives based on a state parameter and sets the task parameter such as goal, timing and selects the motor primitives we want to use. The resulting movement is executed using control law which generates the required motor torques. The teacher provides learning signals on all levels.

A. Skill Representation with MoMP

The MoMP framework is a new approach to build a motor primitive library and select the right movement depending on the external stimuli or create a new elementary movement by mixing several motor primitives that corresponds to this external signal. Here we first describe the dynamic systems motor primitives which are the components of the library. Then, we present how to combine the single motor primitives to select the appropriate motor primitive or to obtain a new movement policy.

1) Dynamic Systems Motor Primitives as Mixture Components: dynamic systems motor primitives (DMP), as suggested by Ijspeert [1], [16], are a particular kind of dynamic systems that is well-suited for 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

$$\dot{z} = h(z). \tag{1}$$

Intuitively, one could say that the canonical systems drives the transformed system. The transformed system

$$\dot{y} = b(y, z, \boldsymbol{w}), \tag{2}$$

is a function of the internal state y, the phase variable z and internal parameters $\boldsymbol{w}$. The formulation of the DMP allows us to represent arbitrarily shaped smooth movements by $\boldsymbol{w}$ which can be estimated by locally weighted regression (see Section II-B.1).

For discrete movements, i.e., movements between fixed start and end points like reaching, pointing, grasping and striking movements, the canonical system can be chosen as

$$\tau \dot{z} = \alpha_z z, \tag{3}$$

where τ is a time constant and α_z is a pre-determined constant parameter which is chosen such that the system is stable [1], [16]. For hitting movements, the appropriate transformed system can be expressed by

$$\begin{aligned} \tau \dot{v} &= (1-z)\alpha_y \left(\beta_y(g_m - y) + \dot{g} - \dot{y}\tau\right) + \eta f(z), \\ \tau \dot{y} &= v, \qquad\qquad g_m = g_m^0 - \dot{g}\tau \frac{\ln(z)}{\alpha_h}, \end{aligned} \tag{4}$$

where y and v are the position and velocity of the system, $\eta = (g_f - y_0)a$, y_0 is the start position, g_f and $\dot{g}$ are the final position and the final velocity of the system, g_m is a moving target and g_m^0 is the initial position of the moving target [9]. Here, α_y and β_y are system parameters which are chosen such that the system is critically damped, a is an amplitude modifier and f is a transformation function which is given by

$$f = \frac{\sum_i w_i \psi_i(z) z}{\sum_i \psi_i(z)}, \tag{5}$$

where $\psi_i(z) = \exp(-\rho_i(z - \mu_i)^2)$ is a Gaussian basis function characterized by a center μ_i and a bandwith ρ_i, see [1], [16]. As z (which has a similar function as a clock) converges to zero at the end of the movement, the influence of the non-linear function f will vanish and the system has only goal g_m as an equilibrium point. We will refer to the acceleration yielded by the DMP as π.

2) Composition with MoMP: Assume we have acquired a skill library consisting of c motor primitives. For each of these motor primitives π_i we save the corresponding external signal $\boldsymbol{x}_i$ which is a part of its goal parameters. For a new external stimulus x, we calculate the similarity to the external stimuli $\boldsymbol{x}_i$ of the motor primitive π_i in the library. The new motor primitive $\pi(\boldsymbol{x})$ is then determined by computing the weighted average of all motor primitives π_i where the weight is defined by a gating network. Thus, we obtain

$$\pi(\boldsymbol{x}) = \frac{\sum_{i=1}^{c} \pi_i \gamma_i k(\boldsymbol{x} - \boldsymbol{x}_i)}{\sum_{i=1}^{c} \gamma_i k(\boldsymbol{x} - \boldsymbol{x}_i)}, \tag{6}$$

where γ_i are weight parameters which can be used to prioritize certain motor primitives, and $k(\boldsymbol{x} - \boldsymbol{x}_i)$ is the kernel which defines the contribution of the motor primitive π_i to the resulting motor policy π (see Figure 2).

In this way, we can both select and generalize the corresponding motor primitives to synthesize a new movement. The resulting policy is based on motor primitives with the most similar input signal $\boldsymbol{x}_i$ that is most similar to the current stimulus $\boldsymbol{x}$. An implementation of the MoMP movement generation is presented in Algorithm 1. Note, that θ, $\dot{\theta}$ and $\ddot{\theta}$ denote the joint position, velocity and acceleration.

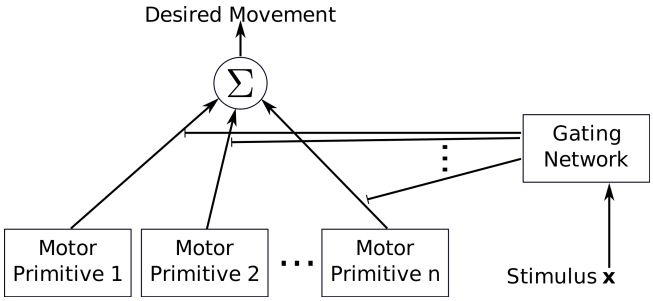


Fig. 2. An illustration of the mixture of motor primitives. The gating network weights the single motor primitives based on the external signal. The weighted sum of these primitives defines the movement.

Algorithm 1 Mixture of Motor Primitives (MoMP)

Input: stimulus $\boldsymbol{x}$
for $t = 1$ to T **do**
  for $i = 1$ to c **do**
    Compute the phase variable z
      $\dot{z}_t = \alpha_z z_t$
    and the transformed system
      $\tau\dot{v} = (1 - z_t)\alpha_\theta \left(\beta_\theta(g_m - \theta) + \dot{g} - \dot{\theta}\tau\right)$
        $+(g_f - \theta_0)af_i(z_t)$
      $\tau\dot{\theta}_i = \dot{v}$
      $g_m = g_m^0 - \dot{g}\tau\frac{\ln(z)}{\alpha_h}$
  end for
  Mixture of motor primitives output
    $\pi(x) = \frac{\sum_{i=1}^{c} \theta_i\gamma_i k(\boldsymbol{x}-\boldsymbol{x}_i)}{\sum_{i=1}^{c} \gamma_i k(\boldsymbol{x}-\boldsymbol{x}_i)}$
end for

B. Learning Skills with MoMP

To generate the motor primitive library we rely on demonstrated movements. Therefore, we have to learn the internal parameters $\boldsymbol{w}$ of the DMP as well as the gating network parameters. To improve the quality of the single motor primitives, the quality of the gating network and to eliminate bad demonstrations from the library, we use the reinforcement learning approaches described in Sections II-B.2 and II-B.3.

1) Imitation Learning for Component Initialization: To build up the library of the different motor primitives, we rely on imitation learning. Imitation learning allows us to learn policies from a given demonstration (for example, by a teacher) and reproduce the movement. We assume that we can learn each primitive separately. Redundant primitives can be eliminated [17].

Assume we have recorded a movement m for one DoF consisting of joint angles θ_t, velocities $\dot{\theta}_t$, and accelerations $\ddot{\theta}_t$ over a time interval $t \in \{1, ..., T\}$. Integrating the canonical system (Equation 3), we can compute a reference transformation function based on Equation (4) by

$$f_t^{\text{ref}} = \frac{\tau^2\ddot{\theta}_t - (1 - z)\alpha_y(\beta_y(g_m - \theta_t) + \dot{\theta}_T - \tau\dot{\theta}_t)}{(\theta_T - \theta_1)a}, \quad (7)$$

where g_m has to be computed in each time step according to Equation (4). To determine $\boldsymbol{w}$, we have to minimize the squared error

$$e_i^2 = \sum_{t=1}^{T} \psi_i(f_t^{\text{ref}} - z_t w_i)^2, \quad (8)$$

which can be solve with locally weighted regression as in [1], [18]. The detailed algorithm for learning motor primitives with imitation learning for one DoF is shown in Algorithm 2.

2) Primitive Adaptation by Trial and Error: Imitation learning suffices for initializing the internal parameters $\boldsymbol{w}$. For subsequent self-improvement of one DMP, we can adapt the motor primitives parameters $\boldsymbol{w}$ using the Policy Learning by Weighting Exploration with the Return (PoWER) algorithm [19]. PoWER is an EM-inspired policy learning method that has been successfully applied to learning single dynamic system motor primitive policies. To update the parameters $\boldsymbol{w}$ with PoWER, we use

$$\boldsymbol{w}_i^{N+1} = \boldsymbol{w}_i^N + \frac{\sum_{j=1}^{N} \boldsymbol{\varepsilon}_i^j \frac{k(\boldsymbol{x}_j - \boldsymbol{x}_i)}{\sum_{l=1}^{c} k(\boldsymbol{x}_j - \boldsymbol{x}_l)} r^j}{\sum_{j=1}^{N} r^j}, \quad (9)$$

where $\boldsymbol{\varepsilon} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{\Sigma})$ denotes the normally distributed exploration, $\boldsymbol{\Sigma}$ is meta-parameter that can also be optimized, r_i is the reward and N is the number of samples. Note, this algorithm enables us to improve one single DMP.

3) Reinforcement Learning for Responsibility Adaptation: Any large set of demonstrations will include attempts that went badly. To make matters worse, the combination of a set of good demonstrations may not necessarily result in good behavior, e.g., when two motor primitives circumvent an obstacle, their weighted combination is likely to drive right into it. Hence, the primitives need to be pruned and adjusted using reinforcement learning.

We created a reinforcement learning algorithm which was inspired by [20]. The responsibility parameter γ_i in Equation (6) is initialized with 1 and is updated according to its contribution to successful and un-successful movements according to

$$\gamma_i^{N+1} = \exp\left(\kappa\frac{s_i - n_i}{\zeta n_i + 1}\right), \quad (10)$$

where

$$\begin{aligned} n_i &= \frac{\sum_{t=1}^{N} \gamma_i^N k(\boldsymbol{x}_t - \boldsymbol{x}_i)}{\sum_{j=1}^{c} \gamma_j^N k(\boldsymbol{x}_i - \boldsymbol{x}_j)}, \\ s_i &= \frac{\sum_{t=0}^{N} \gamma_i^N k(\boldsymbol{x}_t - \boldsymbol{x}_i) r_t}{\sum_{j=1}^{c} \gamma_j^N k(\boldsymbol{x}_i - \boldsymbol{x}_j)}, \end{aligned} \quad (11)$$

indicates the amount of how often the motor primitive i is used and how often it is successfully used, respectively. The parameter κ is the counterpart to a learning rate, ζ is a trade-off parameter between exploration and exploitation, $r \in [0, 1]$ is a reward and N is the number of the executed sample.

Simply put, we re-weight the contribution of motor primitives according to their capability to produce successful movements and even to remove bad demonstrations from the library. Both, the adaptation of one single motor primitive and the adaptation of the gating network to select and combine single DMPs are represented in Algorithm 3.

Algorithm 2 Imitation Learning of one DMP for MoMP

Input: $m_i = (\theta_t, \dot{\theta}_t, \ddot{\theta}_t)$, $t \in \{1, ..., T\}$ for one DoF
Compute $\boldsymbol{w}_i$ using locally weighted regression.
  Determine amplitude a.
  Compute f^{ref}
    $f_t^{\text{ref}} = \frac{\tau^2\ddot{\theta}_t - (1-z)\alpha_y(\beta_y(g_m - \theta_t) + \dot{\theta}_T - \tau\dot{\theta}(j))}{(\theta_T - \theta_1)a}$
  Compute diagonal matrix $\Psi = \text{diag}(\psi_1, ..., \psi_M)$.
  Compute $\boldsymbol{Z} = \{z_1, ..., z_T\}$.
  Compute weights
    $\boldsymbol{w}_i = (\boldsymbol{Z}^T\boldsymbol{\Psi}\boldsymbol{Z})^{-1}\boldsymbol{Z}^T\boldsymbol{\Psi}\boldsymbol{f}_{\text{ref}}$

Algorithm 3 Primitive and Responsibility Adaptation

Initialize $\gamma_i^1 = 1$, $s_i^1 = 0$, $n_i^1 = 0 \ \ \forall i \in \{1, ..., c\}$
repeat
  Get stimulus $\boldsymbol{x}$
  Compute $\pi(\boldsymbol{x})$ using MoMP with parameter $\boldsymbol{w}_i^N$ for all mixture components $i \in \{1, ..., c\}$.
  Perform movement given by $\pi(\boldsymbol{x})$
  Compute reward r^N
  Set $\varepsilon_i^N \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}) \ \ \forall i$
  for each DMP i contributing to $\pi(\boldsymbol{x})$ **do**
    Update weights for DMP

$$\boldsymbol{w}_i^{N+1} = \boldsymbol{w}_i^N + \frac{\sum_{j=1}^N \boldsymbol{\varepsilon}_i^j \frac{k(\boldsymbol{x}_j - \boldsymbol{x}_i)}{\sum_{l=1}^c k(\boldsymbol{x}_j - \boldsymbol{x}_l)} r^j}{\sum_{j=1}^N r^j}$$

    Update responsibility parameter γ_i^{N+1}

$$s_i^{N+1} = s_i^N + \frac{\gamma_i^N k(\boldsymbol{x}_i - \boldsymbol{x}) r_t}{\sum_{j=1}^c \gamma_j^N k(\boldsymbol{x} - \boldsymbol{x}_j)}$$

$$n_i^{N+1} = n_i^N + \frac{\gamma_i^N k(\boldsymbol{x}_i - \boldsymbol{x})}{\sum_{j=1}^c \gamma_j^N k(\boldsymbol{x} - \boldsymbol{x}_j)}$$

$$\gamma_i^{N+1} = \exp\left(\kappa \frac{s_i - n_i}{\zeta n_i + 1}\right)$$

    if $\gamma_i^{N+1} \leq \gamma_{\max}$ is below a threshold $\gamma_{\max}$
      Delete motor primitive i from library
    end if
  end for
until Converges $\boldsymbol{w}^{N+1} \approx \boldsymbol{w}^N$ and $\gamma^{N+1} \approx \gamma^N$

C. Learning the Task Parameters

The MoMP has open task parameters that are given by the supervisor. Striking movements are likely to be similar for many hitting sports and may be adapted straightforwardly from table tennis to real tennis. However, the task parameters may be vastly different.

To return an incoming table tennis ball to the opponent's court, we have to determine when and where the robot has to hit the ball to select the corresponding motor primitive with MoMP. Furthermore, we need to know which joint angles and joint velocities are required at the virtual hitting point to determine the goal parameters of the hitting primitives. Both problems will be adressed in the following. First, we describe how to predict the virtual hitting point, i.e., the position and time at which to hit the ball. In this context, we also predict the velocity of the ball at the virtual hitting point which is part of the external signal for the MoMP. Second, we learn the appropriate mapping to configuration space, i.e., the joint angles and velocities of the virtual hitting point.

1) Predicting the Virtual Hitting Point: For stroke movement generation, we rely on the theory that humans identify virtual targets [21], i.e., the hitting point $\boldsymbol{p}_{\text{hit}} = [p_x, p_y, p_z]^T$, the velocity of the ball $\boldsymbol{v}_{\text{hit}} = [v_x, v_y, v_z]^T$ and the time to contact t_{hit} are already determined when the movement is initiated. To predict these parameters, we use Gaussian process regression (GPR) with a Gaussian kernel based on vision information on the ball's movement.

The prediction is based on the position and velocity of the ball at four time points after passing a virtual plane s. The output is given by the point where the ball is hit by the racket, the velocity of the ball, and the hitting time at ball-racket impact. Since the dimensions of the hitting point and the ball velocity are independent, we have seven independent output values and can model each of these seven parameters independently with GPR models.

2) Learning the Stimuli to Configuration Space Mapping: To generate the arm trajectories, we have to determine the constraints, i.e., start and end position, velocity and acceleration, for the movements on each joint of the arm in each stage. While desired configurations suffice for the awaiting, preparation and follow through stages, the final joint configuration of the hitting stage depends on the task parameters which are determined during the stroke. Thus, we have to update the hitting configuration consisting of joint state $\boldsymbol{\theta}_{\text{hit}}$ and joint velocity $\dot{\boldsymbol{\theta}}_{\text{hit}}$ efficiently but also compatible with the executed movement.

We decided to learn to predict the joint configuration ($\boldsymbol{\theta}_{\text{hit}}$, $\dot{\boldsymbol{\theta}}_{\text{hit}}$) at the intersection point. Unfortunately, there exist multiple solutions since we have a redundant system and not all of them are compatible with the movement. This part may become problematic for a supervised learning system as it would average over a non-convex combination of joint configurations.

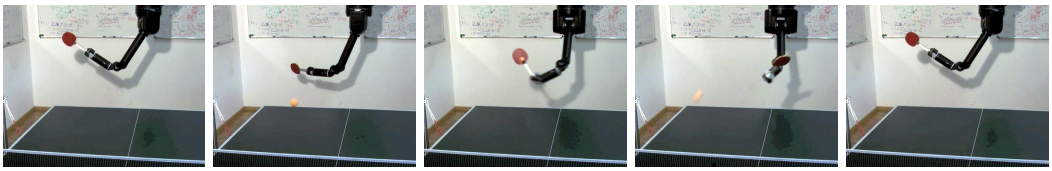
To deal with this non-convexity issue, we learn the mapping using Cost-regularized Kernel Regression as suggested in [22]. This approach forces the learning system to choose a particular kind of solution among the different incompatible possibilities, as we do not trust solutions that differ too much from the typical hitting posture. Since we want to have a human-like striking motion, we search for a solution that looks comfortable for the human observer. Therefore, we define a hitting comfort posture of the arm [23]. The cost function for the Cost-regularized Kernel Regression is given by the sum of the squared error of the joint angles and the hitting comfort posture.

III. Evaluation

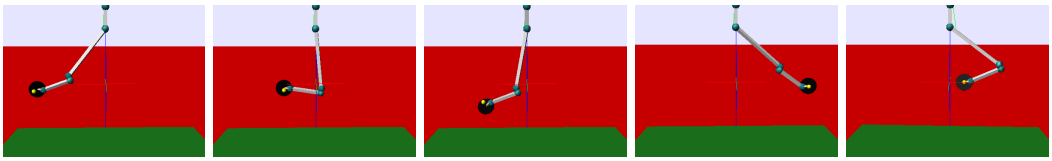
Here, we evaluate the presented concept of MoMP using hitting movements executed by an analytical table tennis robot as described in [24]. Therefore, we give an overview of the table tennis task and evaluate the setup for learning the context parameters for motor primitives, compare the results with the analytical player and show that the MoMP framework can be used to learn a hitting movement and that the setup enhances the performance of the player.

A. Table Tennis Task

For the table tennis task, we developed an environment using the SL framework [25] consisting of a real-world setup and a sufficiently realistic simulation. The setup includes a Barrett WAM arm with seven DoFs that is capable of high speed motion and a vision system with four 200 Hz Procsilica Gigabit GE640C cameras. A racket is attached to the end-effector. Table, racket and ball are compliant with the international rules of human table tennis. The ball is served by a ping pong ball launcher to the forehand of the robot with a randomly chosen velocity covering an area



(a) A series of frames depicting a forehand of the Barrett WAM arm using hitting primitives [19].



(b) Generalization to various targets with a set of primitives from MoMP.

Fig. 3. A sequence of the Barrett WAM arm striking table tennis balls using hitting primitives [19] and the generalization of hitting a table tennis ball.

of approximately $1\,\mathrm{m}^2$. The ball is visually tracked with a sampling rate of 60 Hz. We use a simplified model of the flight and bouncing behavior of the ping pong ball, i.e., we consider gravity but neglect the air drag and spin acting on the ball. The coefficients of restitution of both racket-ball and ball-table interactions were determined empirically.

If the system detects a ping pong ball that is moving towards the robot the preparation stage is initiated and the goal parameters, i.e., the time and position until intersection as well as the velocity of the ball at impact time is estimated. The racket of the player moves backwards in order to prepare the stroke. The preparation stage is executed until the estimated time to impact is smaller or equals the time for the hitting stage. In the hitting stage the racket moves towards the virtual hitting point until it hits the ball in a circular movement with a specific velocity. After the hitting stage is completed, the follow through stage is executed where the racket continues to move upwards with decreasing velocity. In the awaiting stage the arm moves back to a default position where the system is waiting until a new ball is detected.

B. Evaluation of the Task Parameter Model

To evaluate the models for the task parameters we compute the normalized mean squared error (nMSE) defined as the ratio of the mean squared error (MSE) and the variance of the target, and investigate the performance of the resulting player in the simulated environment. For all evaluations, the ball was served 10,000 times.

1) Modeling the Virtual Hitting Point: To estimate the hyperparameters and train each model, we used 1000 observed input-output pairs of an analytical table tennis player [24]. All seven models were trained offline.

The GPR model was evaluated with a data set consisting of 3500 trials. The MSE of the position estimation are below 1 mm. The MSE of the estimated velocity is below 2 mm/s and of the time estimation below 1 ms. The nMSE of all models is below 0.01. The performance is identical with the performance of a manually programmed player which uses a physical model to predict the goal parameters.

2) Learning the Hitting Configuration: To learn a model to map the movement stimuli to configuration space, we use 2500 input-output pairs of a manually programmed table tennis player. For each DoF we learn the corresponding joint angles and velocities offline in an independent model. From the resulting system we sampled again 2500 input-output pairs to train the model again. We repeated this policy iteration until the policy converges.

The model from the first iteration is able to return 94% of the balls and 48% of the balls successfully to the opponent's court. We used the data generated with this player to train the model again. The improved model had a significantly improved performance. The player was able to return 95% of the balls to the opponent's court where 63 % hit the desired location. Further iterations did not improve the performance as the nMSE for all joint angles was already below 0.01. The failures to return the ball successfully are the result of an inaccurate model for the joint velocities of the wrist joints.

C. Learning the Motor Task using MoMP

The movements for all four stages were generated using DMPs as described in Section II-A.1. For the awaiting, preparation and follow through stage we used single motor primitives. Trajectories of these movements were sampled from an analytical table tennis player. The parameters of the movement of the hitting stage depends on the task parameters and vary in their overall shape. To generate a movement that is able to cope with the varying conditions, we use the MoMP with the virtual hitting point and the estimated ball velocities as external signal. Therefore, we built a movement library for the stroke consisting of 300 motor primitives sampled from successful strokes of the analytical player. We collected arm, racket and ball trajectories and extracted the duration of the stages and the Cartesian ball positions and velocities at the hitting point. The parameters $\boldsymbol{w}$ of all motor primitives are learned offline by imitation learning as described in Algorithm 2. All DoFs are modeled independently in the transformed system but are synchronous as they start all at the same time, have the same duration and are driven by the same canonical system.

First, we evaluated the single mixture components (i.e., the motor primitives learned by imitation learning) alone. The performance varied as much as the quality of the used

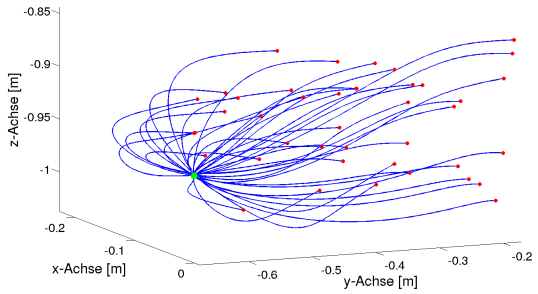


Fig. 4. Trajectories for the end-effector of the Barrett WAM in the hitting stage. Starting point and goals are marked with a green and red dot, respectively.

demonstration for the motor primitive for the hitting stage. Testing selected motor primitives, we observed a success rate between 23 % – 89 %. We combine these components in a mixture of motor primitives with initial weights of $\gamma_i = 1$ for the hitting stage. The resulting player resulted in 67 % successful returns. Learning the weight parameters γ_i as suggested in Section II-B.3 resulted in an improved performance of 94 %. Analyzing the weight parameters for all motor primitives allowed understanding the results as all primitives i where γ_i converged to zero were effectively removed from the policy. We observed that about 27 % of the primitives were removed and that these primitives had an average performance of 30 %.

IV. Conclusion and Future Work

We presented a novel approach for using motor primitives for complex tasks in a MoMP framework. We evaluated the approach in a table tennis setup where we were able to improve the performance of the player. Furthermore, we learned the goal parameters used to select and initiate the appropriate motor primitives. This step included a learned model for the inverse kinematics of a redundant robot arm. As a result, we obtain a fully learned robot system that is able to return balls served randomly with a ball launcher to the opponent's court.

Note, this framework is more challenging as previous work on robot table tennis [11], [12], [26] as it does not use smart engineering to overcome inherent problems like movement generation and the orientation of the racket. In contrast to these approaches, we use an anthropomorphic robot arm with seven DoFs. We concentrate on learning smooth movements that properly distribute the forces over the different DoFs yielding an human like movement pattern.

Implementing the table tennis framework on an humanoid robot would enable the system to move sideways as well as forward and backward. These additional DoFs would enable the system to apply the motor primitives for different positions without drastically changing the overall shape for the arm movement. However, applying the framework on an humanoid robot has the consequence that we have to deal with the balance of the system, the coordination of the additional DoFs and the increasing complexity of the strategy in table tennis.

In future work we will integrate a strategy using context information and thus, select a specific striking movement that returns the ball such that the opponent cannot return it.

References

[1] A. J. Ijspeert, J. Nakanishi, and S. Schaal, "Movement imitation with nonlnear dynamical systems in humanoid robots," in *Proc. Int. Conf. on Robotics and Automation (ICRA)*, 2002.

[2] S. Schaal, J. Peters, J. Nakanishi, and A. Ijspeert, "Control planing, learning and imitation with dynamic movement primitives," in *Proc. Workshop Bilateral Paradigms on Humans & Humanoids IROS*, 2003.

[3] J. Nakanishi, J. Morimoto, G. Endo, G. Cheng, S. Schaal, and M. Kawato, "Learning from demonstration and adaption of biped locomotion," *Robotics and Autonomous Systems (RAS)*, vol. 47, no. 2-3, pp. 79 – 91, 2004.

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

[5] F. Guenter, M. Hersch, S. Calinon, and A. Billard, "Reinforcement learning for imitating constrained reaching movements," *Advanced Robotics, Special Issue on Imitative Robots*, vol. 21, no. 13, 2007.

[6] J. Kober, B. Mohler, and J. Peters, "Learning peceptual coupling for motor primitives," in *Proc. Int. Conf. on Intelligent RObots and Systems (IROS)*, 2008.

[7] V. Zordan and J. Hodgins, "Motion capture-driven simulations that hit and react," in *SIGGRAPH Symp. on Computer Animation*, 2002.

[8] E. Aboaf, S. Drucker, and C. Atkeson, "Task-level robot learning: Juggling a tennis ball more accurately," in *Proceedings of IEEE International Conference on Robotics and Automation*, 1989.

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

[10] U. Frese, B. Bäuml, S. Haidacher, G. Schreiber, I. Schaefer, M. Hähnle, and G. Hirzinger, "Off-the-shelf vision for a robotic ball catcher," in *Proc. Int. Conf. on Intelligent Robots and Systems*, 2001.

[11] R. Andersson, *A robot ping-pong player: experiment in real-time intelligent control*. Cambridge, MA, USA: MIT Press, 1988.

[12] H. Fassler, H. Vasteras, and J. Zurich, "A robot ping pong player: optimized mechanics, high performance 3d vision, and intelligent sensor control," *Robotersysteme*, pp. 161–170, 1990.

[13] A. Gams and A. Ude, "Generalization of example movements with dynamic systems," in *Humanoids*, 2009.

[14] T. Matsubara, S. Hyon, and J. Morimoto, "Learning stylistic dynamic movement primitives from multiple demonstrations," in *Proc. Int. Conf. on Intelligent RObots and Systems*, 2010.

[15] M. Jordan and R. Jacobs, "Hierarchical mixures of experts and the em algorithm," *Neural Computation*, vol. 6, pp. 181 – 214, 1994.

[16] S. Schaal, P. Mohajerian, and A. Ijspeert, "Dynamics systems vs. optimal control – a unifying view," *Progress in Brain Research*, vol. 165, no. 1, pp. 425 – 445, 2007.

[17] S. Chiappa, J. Kober, and J. Peters, "Using bayesian dynamical systems for motion template libraries," in *Advances in Neural Information Processing Systems 22 (NIPS)*, 2008.

[18] S. Schaal, J. Peters, J. Nakanishi, and A. Ijspeert, "Learning motor primitives," in *International Symposium on Robotics Research*, 2003.

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

[20] J. Peters, K. Muelling, and Y. Altun, "Relative entropy policy search," in *Proceedings of the Twenty-Fourth National Conference on Articial Intelligence (AAAI-10)*, 2010.

[21] M. Ramanantsoa and A. Durey, "Towards a stroke construction model," *International Journal of Table Tennis Science*, vol. 2, 1994.

[22] J. Kober, E. Oztop, and J. Peters, "Reinforcement learning to adjust robot movements to new situations," in *R:SS*, 2010.

[23] H. Cruse, M. Brüwer, P. Brockfeld, and A. Dress, "On the cost functions for the control of the human arm movement," *Biological Cybernetics*, vol. 62, pp. 519–528, 1990.

[24] K. Muelling and J. Peters, "A computational model of human table tennis for robot application," in *Autonome Mobile Systeme 2009*, 2009.

[25] S. Schaal, "The SL simulation and real-time control software package," Tech. Rep., in preparation.

[26] F. Miyazaki, M. Matsushima, and M. Takeuchi, "Learning to dynamically manipulate: A table tennis robot controls a ball and rallies with a human being," in *Advances in Robot Control*. Springer, 2005.