Learning to Select and Generalize Striking Movements in Robot Table Tennis

Katharina Mülling[1,2], Jens Kober[1,2], Oliver Kroemer[2], Jan Peters[1,2]
email: {*muelling,kober,jan.peters*}*@tuebingen.mpg.de*
oli@robot-learning.de

(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 from physical interactions is an important goal for both robotics and machine learning. However, when moving beyond basic skills, most monolithic machine learning approaches fail to scale. For more complex skills, methods that are tailored for the domain of skill learning are needed. In this paper, we take the task of learning table tennis as an example and present a new framework that allows a robot to learn cooperative table tennis from physical interaction with a human. The robot first learns a set of elementary table tennis hitting movements from a human table tennis teacher by kinesthetic teach-in, which is compiled into a set of motor primitives represented by dynamical systems. The robot subsequently generalizes these movements to a wider range of situations using our mixture of motor primitives 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

Humans perform complex motor tasks in uncertain and changing environments with little apparent effort. They are able to generalize and, as a consequence, to adapt to new tasks based on their motor abilities. In contrast, current robots rely strongly on well-modeled environments with accurately estimated parameters. Therefore, changes in the environment or task require an expert to program new rules of motor behaviors. To alleviate this problem, robots need to autonomously learn new motor skills and improve their abilities. While this problem has been recognized by the robotics community (Schaal et al., 2002), it is far from being solved. Instead, learning new motor tasks autonomously and adapting motor skills online while interacting with the environment has become 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 (Giszter et al., 2000; Billard et al., 2008; Flash and Hogan, 1985). 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 (Guenter et al., 2007; Peters and Schaal, 2008b; Kober et al., 2008; Peters and Schaal, 2008a; Bitzer et al., 2010). 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, 2008b,a; Bitzer et al., 2010) or through the use of a motion capture system such as a VICON setup (Kober et al., 2008). We refer to (Schaal et al., 2003a; 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 to the requirements of the current task, several methods have been suggested. Several approaches make use of via-point based formulations using splines to interpolate between the via-points (Miyamoto et al., 1996), Hidden Markov Models (HMMs) (Williams et al., 2008) or Gaussian Mixture Models (GMMs) (Calinon et al., 2007). While these approaches are favorable in many situations, spline-based via-point models are difficult to use in new situations while HMMs und GMMs are difficult to train for high-dimensional systems. Hence, an alternative approach based on dynamical systems was suggested by Ijspeert et al. (2002) and called DMPs (Dynamical system Motor Primitives). DMPs are robust against perturbations, allow changing the final state, speed, and duration of the motion without altering the overall shape of the movement. Furthermore, they are straightforward to learn by imitation learning (Ijspeert et al., 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 (Schaal et al., 2003b; Nakanishi et al., 2004), tennis-like swings to a static end-point (Ijspeert et al., 2002), T-ball batting (Peters and Schaal, 2006), constrained reaching tasks (Guenter et al., 2007), and Ball-in-a-cup (Kober et al., 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), to improve the performance of the stored movements with respect to the current situation as well as to select and generalize 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. Our approach is validated using robot table tennis as a benchmark task. In table tennis, the robot has to interact in real-time with a human partner. Thus, it has to adapt to the humans variable behavior. A typical movement in table tennis consists of the preparation of the stroke by moving backwards, hitting the ball at a desired position, with the right orientation and velocity and moving the arm back to a rest posture. The hitting movement itself 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 can result in unsuccessful attempts to return the ball to the opponent's court. The goal of this physical human-robot interaction 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. Thus, the system recorded the demonstrated movement of the arm and the position of the ball. From this stream of data, the movement primitives were extracted. Secondly, the learning system needs to identify the augmented state that includes where, when and how the ball should be hit. Subsequently, the system generalizes

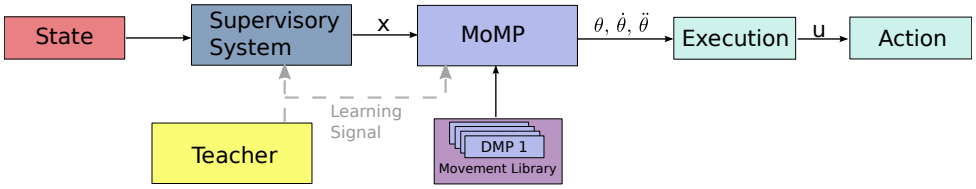


Figure 1: General setup for learning a motor task using the mixture of motor primitives (MoMP). A supervisory level creates the augmented state $\mathbf{x}$ containing the relevant information of the task based on the state of the system. MoMP selects and generalizes among the movement templates in a library according to the augmented state $\mathbf{x}$ which is provided by the supervisory level. 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 such that the system is able to adapt both the generation of the task relevant information and the generation of the movement.

these movements to a wider range of situations using the proposed MoMP algorithm. Here, generalizing refers to the ability to generate striking movements for an arbitrary incoming ball that was not seen during the demonstrations. The resulting system is able to return balls served by a ball launcher as well as to play in a match against a human.

In the remainder of the paper, we will proceed as follows. In Section 2, we present our general framework for learning complex motor skills. We show (i) how to initialize the library using imitation learning, (ii) how to derive the augmented state from the state of the system and (iii) how to adapt the setup to new situations using the MoMP algorithm. We evaluate the MoMP approach in a robot table tennis scenario in Section 3. Here, we will use all components of the presented motor skill learning framework to achieve this task. In Section 4, we will present our results and summarize our approach. A glossary with the definition for the technical terms used in this paper can be found in Appendix B. A review of robot table tennis is presented in Appendix F. The overall performance of our approach is illustrated in the attached video (Extension 1).

2 Learning and Generalizing 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, 2008a; Pongas et al., 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 small number of movement primitives. Hence, we propose an algorithm called mixture of motor primitives (MoMP) 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 states are extracted from a set of demonstrations and 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.

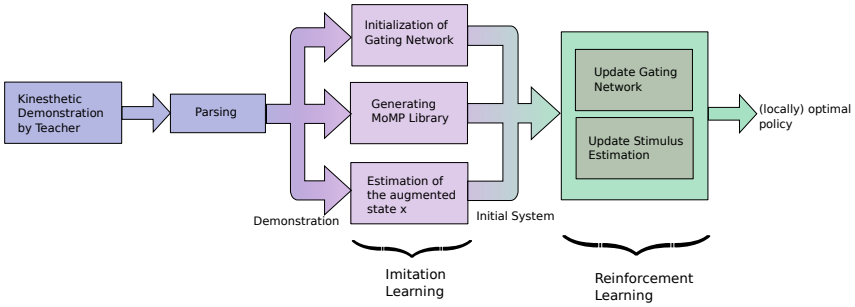


Figure 2: The learning process in the presented MoMP framework. Given a set of demonstrations recorded by kinesthetic teach-in, the system generates the movement library, initializes the gating network as well as the estimation of the augmented states used in the MoMP. The resulting initial system can be used to perform a given motor task. During the execution, the gating network, the augmented states estimation and the primitives are further improved using reinforcement learning.

The supervisory system which generates the augmented state, as well as the gating network of the MoMP algorithm that selects and mixes the movement primitives according to the augmented state 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. The learning structures involved in the MoMP algorithm are shown in Figure 2.

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 Motor Task using the 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 $\mathbf{x} = [\mathbf{s}, \boldsymbol{\delta}]$, 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 (more details on the acquisition of single primitives will follow in Section 2.3.3).

The MoMP generates a new movement for the current situation, represented by the augmented state $\mathbf{x}$ by computing the weighted average of all movement primitives π_i (see Figure 3). The resulting motor policy generated by MoMP is given by

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

where the function $\gamma_i(\boldsymbol{\delta})$ generates the weight of $\pi_i(\mathbf{x})$ given the augmented state $\mathbf{x} = [\mathbf{s}, \boldsymbol{\delta}]$. All weights γ_i together form the *gating network* of the MoMP similar to a gating network in a mixture of experts (Jacobs et al., 1991).

The weights of the gating network ensure that only movement primitives that are well-suited for the current situation contribute to the resulting behavior. It appears reasonable to assume that movement

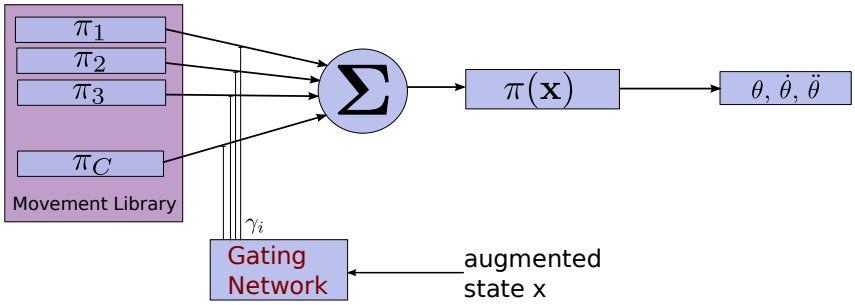


Figure 3: An illustration of the mixture of motor primitive framework. The gating network weights the single movement templates stored in a movement library based on an augmented state. The weighted sum of these primitives defines the new motor policy which produces the joint positions, velocities and accelerations for one degree of freedom. The resulting movement is then executed using a control law for execution which generates the required motor torques.

primitives associated with augmented states 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 more suitable 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

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

where $\phi_i(\mathbf{x})$ denotes the feature vector of $\mathbf{x}$, $\boldsymbol{\theta}_i$ 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 $\boldsymbol{\theta}_i$ 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 $\boldsymbol{\theta}_i$ 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 (see Appendix D for a short review of linear Bayesian regression). 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 Computation of 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,

Algorithm 1 Generalizing Movements

Initialize:

Initiate movement library from demonstrations

Determine initial states $\mathbf{S}^0$, costs $\mathbf{C}^0$ and meta-parameters $\mathbf{D}^0$ from demonstrations

Choose a kernel k, $\mathbf{k}$, $\mathbf{K}$ and scaling parameter λ

Choose basis function $\phi(\mathbf{x})$, reward function r

For each trial

Determine current state $\mathbf{s}^{N+1}$

Choose $\boldsymbol{\delta}^{N+1} \sim \mathcal{N}(\boldsymbol{\delta}|\overline{\delta}(\mathbf{s}), \Sigma(\mathbf{s}))$ with CrKR, where

$\overline{\boldsymbol{\delta}}(\mathbf{s}) = \mathbf{k}(\mathbf{s})^T(\mathbf{K} + \lambda\mathbf{C})^{-1}\mathbf{D}$

$\Sigma(\mathbf{s}) = k(\mathbf{s},\mathbf{s}) + \lambda - \mathbf{k}(\mathbf{s})^T(\mathbf{K} + \lambda\mathbf{C})^{-1}\mathbf{k}(\mathbf{s})$

Set $\mathbf{x}^{N+1} = [\mathbf{s}^{N+1}, \boldsymbol{\delta}^{N+1}]$

Calculate motor policy with MoMP

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

Execute motor policy $\pi(\mathbf{x})$

Determine reward r for executing $\pi(\mathbf{x})$

Update $\mathbf{S}$, $\mathbf{D}$ and $\mathbf{C}$ according to $\mathbf{s}$, $\boldsymbol{\delta}$ and r

For all π_i in library update θ_i with LBR

$c_i = \gamma_i / \sum_j^L \gamma_j$

$\mathbf{V}_i^{N+1} = \left(\left(\mathbf{V}_i^N\right)^{-1} + \beta\phi(\mathbf{x}_{N+1})^T c_i \phi(\mathbf{x}_{N+1})\right)^{-1}$

$\boldsymbol{\theta}_i^{N+1} = \mathbf{V}_i^{N+1}\left(\left(\mathbf{V}_i^N\right)^{-1}\boldsymbol{\theta}_i^N + \beta\phi(\mathbf{x}_{N+1})^T c_i r\right)$

end

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 et al. (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 Kernel Regression (CrKR), see (Kober et al., 2012), has also proven to be suitable for learning meta-parameters for motor policies as used in this project. CrKR uses a stochastic policy from which it draws the meta-parameters. The initial policy is obtained by training the algorithm with the demonstrations. The cost is used to weight the training data points down based on the reward r. See Appendix C for a detailed description of the CrKR algorithm.

In Algorithm 1, we show the complete approach of computing the meta-parameters $\boldsymbol{\delta}$ using CrKR, the generation of a generalized motor policy using the MoMP, and the corresponding update according to the system's performance.

2.3 Representation of Behavior with Movement Primitives

To represent a single motor policy π_i used in the MoMP, we employ motor primitives represented by dynamical systems (DMPs). DMPs, as suggested in (Ijspeert et al., 2002; Schaal et al., 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 determines the 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. The desired movement is specified by a dynamical system for each DoF, linked together by one shared canonical system.

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 (see Section 2.3.3). 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. In contrast to non-autonomous movement representations such as splines, DMPs are robust against perturbations in the environment, because time is not used explicitly. Furthermore, the DMP representation allows us to mimic a presented movement shape and, to represent arbitrary movements instead of only one special kind of movement. 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. Thus, the use of movement primitives for the whole striking movement with zero end-velocity is not suitable. The generated movements from such an approach will fail to return the ball successfully, since the trajectory is not guaranteed to cross the hitting point with the desired end-effector velocity at the right point in time. As a result, the success-rate of returning a ball to the opponent's court would decrease drastically. Hence, we need movement 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 movement primitive suited for hitting movements and use the feature of ball-racket contact to allow the system to switch the stage. Using the ball-based features, it becomes straightforward to compute the hitting point and therefore, the point were we have to switch between the two movement stages. 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, we will show in the following the shortcomings of their approach and present an improved alternative form.

2.3.1 Movement Primitives for Striking Motions

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 as

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

where τ is a temporal scaling factor and α_z is a pre-defined constant which is chosen such that the behavior is stable (Ijspeert et al., 2002; Schaal et al., 2007). Initially z is set to 1 and converges to zero at the end of the movement. Please note that as long as the time parameter of the movement does not change, we can integrate with respect to the time directly and use the solution $z(t) = \exp(-\alpha_z/\tau t)$ instead. However, implementing a change of the hitting time, slowing down or speeding up is not straight forward anymore. Therefore, when such parameter changes are essential (as in our setup), the phase should be computed as described in Equation 5.

For hitting movements, Kober et al. (2010) proposed the transformed system

$$\begin{aligned} \tau \dot{v} &= (1-z)\alpha_y \left(\beta_y(g-y) + \dot{g}_f \tau - v\right) + \eta f(z), \\ \tau \dot{y} &= v, \qquad\qquad g = g^0 - \dot{g}\tau \frac{\ln(z)}{\alpha_z}, \end{aligned} \tag{6}$$

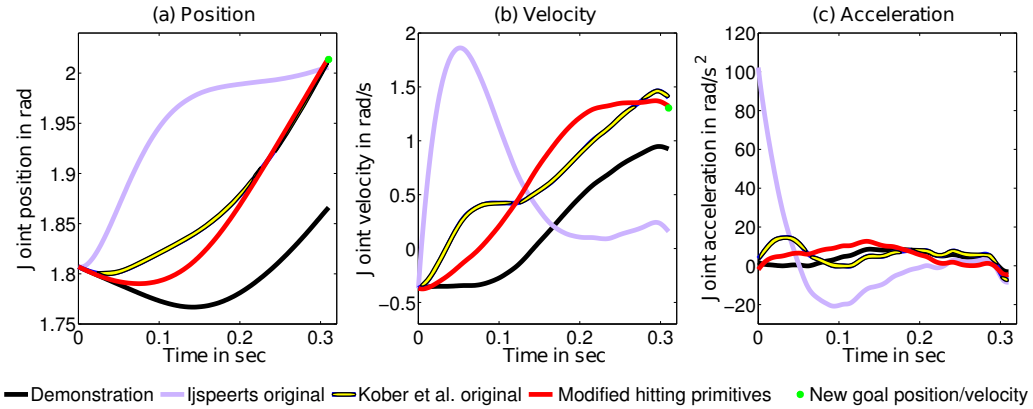


Figure 4: Changing the goal position and velocity is essential for adapting the demonstration to new situations. This figure illustrates how the different versions of the dynamical system based movement primitives are modulated by changing the goal position and velocity of the movement as they frequently occur in striking sports. The demonstration of a striking movement was obtained by kinesthetic teach-in in table tennis. After learning, all movement primitive formulations presented were able to reproduce the demonstration to a certain extend. However, the three formulations are differently robust against changes in the desired goal position and velocity. We changed the position by 0.15 m and the velocity by 0.4 m/s. The original formulation of Ijspeert is not able to reach the desired velocity as the system ends with zero 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 y and v are the desired position and velocity generated by the policy, $\eta = (g_f - y_0)$ defines the amplitude of the movement, y_0 is the start position, g_f is the desired final position, $\dot{g}_f$ the desired final velocity of the system, g is the current goal position defined by a moving target, $g^0 = g_f - \tau \dot{g}_f$ is the initial position of the moving target and $\tau \ln(z)/\alpha_z$ corresponds to the time. At the end of the movement, g is equal to the desired final goal g_f. For each new movement g and $\dot{g}$ are computed based on the new desired goal position g_f and the desired final velocity $\dot{g}_f$. The pre-defined spring-damper constants α_y and β_y are chosen such that the system 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. If $\dot{g}$ is set to zero, the formulation corresponds to the original formulation of Ijspeert except for the first term $(1 - z)$ which prevents large jumps in the acceleration at the beginning of the movement. A proof of the stability of the system can be found in Appendix E.

2.3.2 Modified Hitting Primitives

The formulation of Kober et al. (2010) with a linear velocity allows to directly incorporate the desired velocity such that the desired final velocity can be adapted (in addition to the desired initial and final position in Ijspeert's formulation). However, this formulation has two drawbacks. First, fast movement

changes of the final position and velocity can lead to inaccuracies in the final velocity (see Figure 4). In table tennis, such errors at the hitting point can cause the returned ball to miss the opponent's court. Second, if $(g_f - y_0)$ is close to zero, the scaling of f with $\eta = (g_f - y_0)$ can cause huge accelerations when g_f is changed (see Figure 5). Using a polynomial target velocity, we modified the transformed system to

$$\begin{aligned}
\tau\dot{v} &= \alpha_y\left(\beta_y(g-y) + \dot{g}\tau - v\right) + \ddot{g}\tau^2 + \eta f(z), \\
\tau\dot{y} &= v, \qquad\qquad \eta = \frac{\exp\left(g_f - y_0\right)}{\exp(a)}, \\
g &= \sum_{i=0}^{5} b_i \left(-\tau\frac{\ln(z)}{\alpha_z}\right)^i, \quad \dot{g} = \sum_{i=1}^{5} i b_i \left(-\tau\frac{\ln(z)}{\alpha_z}\right)^{i-1}, \\
\ddot{g} &= \sum_{i=2}^{5} (i^2 - i) b_i \left(-\tau\frac{\ln(z)}{\alpha_z}\right)^{i-2},
\end{aligned} \tag{8}$$

where g, $\dot{g}$ and $\ddot{g}$ are the current position, velocity and acceleration defined by the moving target 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 fifth 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. Thus, using a fifth order polynomial allows us to control the initial and final position, velocity and acceleration more accurately. The new scaling term $\eta = \exp(g_f - y_0)/\exp(a)$ ensures that f does not cause infeasible acceleration. To avoid infeasible acceleration profiles the velocity and acceleration is set constant outside the interval $[0, T_f]$. The term $(1-z)$ is not necessary anymore, as jumps in the acceleration are avoided by incorporating the polynomial position profile starting at y_0. Note that Ning et al. (2011) also used splines in a DMP setting. However, while Ning et al. (2011) always try to follow the reference trajectory and adapt only the beginning and ending of the trajectory, we attempt to adapt the whole trajectory. A proof of the stability of the system can be found in Appendix E.

The complete algorithm for generalizing DMPs using MoMP is given in Algorithm 2.

2.3.3 Initialization of the Behaviors

A library of movement primitives can be built using demonstrated motor behaviors. Imitation learning for DMPs, as suggested in (Ijspeert et al., 2002; Schaal et al., 2003b), enables us to learn initial DMPs from observed trajectories and to reproduce these movements. The observed trajectories are captured using kinesthetic teach-in. To learn the DMP from a demonstration, we assume that the policy that produced the observed trajectory can be represented by a DMP as described in Equation (8). Consequently, the problem reduces to inferring the set of internal parameters $\mathbf{w}$ such that the error between the reproduced and demonstrated behavior is minimized. Therefore, the problem can be solved straightforwardly by a regression algorithm.

In the following, we will assume that each DoF is independent and that we can learn each primitive separately. Redundant primitives can be eliminated (Chiappa et al., 2008). For a recorded trajectory $\Gamma = [\theta_t, \dot{\theta}_t, \ddot{\theta}_t]$ over a time interval $t \in \{1, ..., T_f\}$, we can compute the reference signal of a striking movement based on Equation (8) by

$$f_t^{\text{ref}} = \tau^2\ddot{\theta}_t - \alpha_y(\beta_y(g - \theta_t) + \tau\dot{g} - \tau\dot{\theta}_t) - \ddot{g}\tau^2. \tag{9}$$

The appropriate cost function to be minimized for each parameter w_n is the weighted squared error

$$e_n^2 \quad = \quad \sum_{t=1}^{T_f} \psi_t^n (f_t^{\text{ref}} - z_t w_n)^2, \tag{10}$$

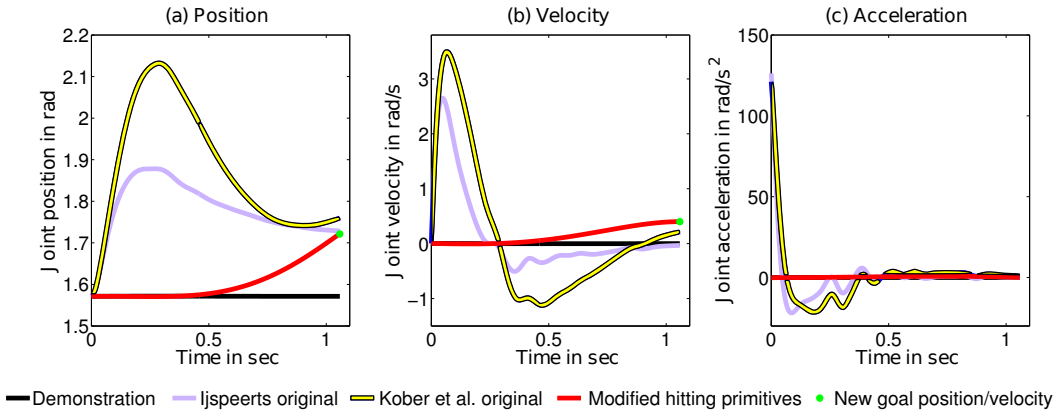


Figure 5: A key problem of the previous movement primitive formulation is the highly uneven distributed accelerations with a peak at the beginning of the movement. Such jumps can affect the position and velocity drastically as shown in this figure. They may result in the attempt to generate infeasible 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. By introducing a new scaling term for f, we can avoid jumps at the beginning of the movement and reduce overshooting in the position and velocity profile. The demonstration was obtained by kinesthetic teach-in of a striking movement. Both position and velocity were then changed to a new goal. Note, that the acceleration of the modified hitting primitives is so much smaller that it appears to be zero when compared to Kober's and Ijspeert's original versions.

where $\psi_t^n = \exp(-\rho_n(z_t - \mu_n)^2)$ while z_t can be determined by integrating Equation (5). We can rewrite this error in matrix form by

$$e_n^2 = (\mathbf{f}^{\text{ref}} - \mathbf{z}w_n)^T \mathbf{\Psi_n} (\mathbf{f}^{\text{ref}} - \mathbf{z}w_n), \tag{11}$$

where $\mathbf{f}^{\text{ref}}$ is the vector consisting of f_t^{ref} for each time step t, $\mathbf{\Psi}_n = \text{diag}\{\psi_1^{\text{n}}, ..., \psi_{\text{T}_\text{f}}^{\text{n}}\}$ and $\mathbf{z} = [z_1, ..., z_{T_f}]^T$. The resulting regression problem can be solved straightforwardly using locally weighted regression as originally suggested in (Ijspeert et al., 2002; Schaal et al., 2003b). Thus, the optimal w_n are given by

$$w_n = (\mathbf{z}^T \mathbf{\Psi}_n \mathbf{z})^{-1} \mathbf{z}^T \mathbf{\Psi}_n \mathbf{f}^{\text{ref}}. \tag{12}$$

The detailed method for learning movement primitives with imitation learning for all DoFs is shown in Algorithm 3.

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

Algorithm 2 Mixture of Motor Primitives (MoMP)

Input: augmented state $\mathbf{x}$
for $t = 1$ to T **do**
  for $i = 1$ to L **do**
    Compute the phase variable z
      $\dot{z} = -\alpha_z z$
    and the transformed system
      $g = \sum_{j=0}^{5} b_j \left(-\tau \frac{\ln(z)}{\alpha_z}\right)^j$
      $\dot{g} = \sum_{j=1}^{5} j b_j \left(-\tau \frac{\ln(z)}{\alpha_z}\right)^{j-1}$
      $\ddot{g} = \sum_{j=2}^{5} (j^2 - j) b_j \left(-\tau \frac{\ln(z)}{\alpha_z}\right)^{j-2}$
      $\eta = \frac{\exp(g_f - y_0)}{\exp a}$
      $\tau \dot{v} = \alpha_y \left(\beta_y (g - y) + \dot{g}\tau - v\right) + \ddot{g}\tau^2 + \eta f(z)$
      $\tau \dot{y} = \dot{v}$
  end for
  Mixture of Motor Primitives output
    $\pi(x) = \frac{\sum_{i=1}^{L} \gamma_i(\mathbf{x}) \pi_i(\mathbf{x})}{\sum_{i=1}^{L} \gamma_i(\mathbf{x})}$
end for

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, i.e., where, when and how the ball has to be hit (see Section 3.2). The target location on the opponent's court was fixed to the center of the court to make the results

Algorithm 3 Imitation Learning of one DMP for MoMP

Input: $\Gamma_i = [\theta_t, \dot{\theta}_t, \ddot{\theta}_t]$, $t \in \{1, ..., T_f\}$
For all DoF i and each parameter w_n **do**
  Set constant parameters α_z, α_y, β_y, p_n and μ_n
  Set $g_f = \theta_\tau$ and $\dot{g}_f = \dot{\theta}_{T_f}$
  Calculate g, $\dot{g}$ and $\ddot{g}$
  Calculate z_t by integrating $\tau \dot{z} = \alpha_z z$ for all t
  Calculate $\psi_t^n = \exp(-\rho_n (z_t - \mu_n)^2)$
  Calculate reference value f^{ref} from demonstration
    $f_t^{\mathrm{ref}} = \tau^2 \ddot{\theta}_t - \alpha_y (\beta_y (g - \theta_t) + \dot{g} - \tau \dot{\theta}_t) - \ddot{g}\tau^2$
  Create matrices $\mathbf{z} = [z_1, ..., z_{T_f}]^T$, $\Psi = \mathrm{diag}(\psi_1^n, ..., \psi_{T_f}^n)$ and $\mathbf{f}^{\mathrm{ref}} = [f_1^{\mathrm{ref}}, ..., f_{T_f}^{\mathrm{ref}}]^T$
  Compute weights via locally weighted regression
    $w_n = (\mathbf{z}^T \mathbf{\Psi} \mathbf{z})^{-1} \mathbf{z}^T \mathbf{\Psi} \mathbf{f}_{\mathrm{ref}}$
end

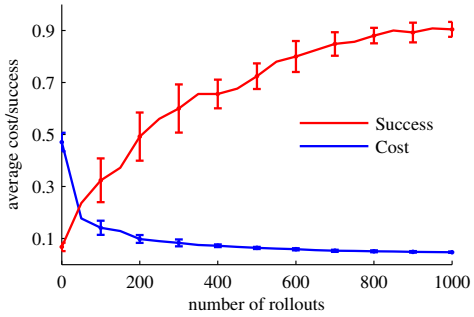


Figure 6: Cost (lower curve) of the simulated table tennis task averaged over 10 runs. The upper curve represents the rate of successful hits. At the beginning the robot misses the ball in 95% of the trials. At the end the trained robot hits almost all balls.

comparable.

Special attention was given to prevent physical damage to the setup during movement execution. As a result, the system evaluates at each time point the position of the end-effector. If the end-effector was too close to the table, the movement would be stopped and the arm moves back to its rest posture. During the real robot evaluation, however, the table was 80 cm away from the real robot. Thus, the robot was not able to hit the table. However, even in simulation, where the table was at a distance of 30 cm from the robot, collisions with the table were rarely a problem. Balls which cannot be reached by the robot are ignored. To avoid joint limitations, the solutions for the joint configuration predicted for the hitting point were evaluated before performing the motion. If the joint configuration violated the joint limits, the solution would be rejected. Furthermore, the desired joint configuration for the next time step, was tested. If the join limitations were violated, the movement would stop and not continue the planned movement. However, if additional object or joint limit avoidance is necessary, the movement could be adapted by adding a repellent force. In this case, a potential field centered around the obstacle (as described by Park et al. (2008) and Kroemer et al. (2010)) could be used. Luckily, these security precautions were never required in our experiments in practice as the MoMP generates movements within the convex combination of demonstrations. Hence, the system will not encounter joint limits or hit the table. Similarly, singularities were avoided by additionally restricting the hitting area of the robot.

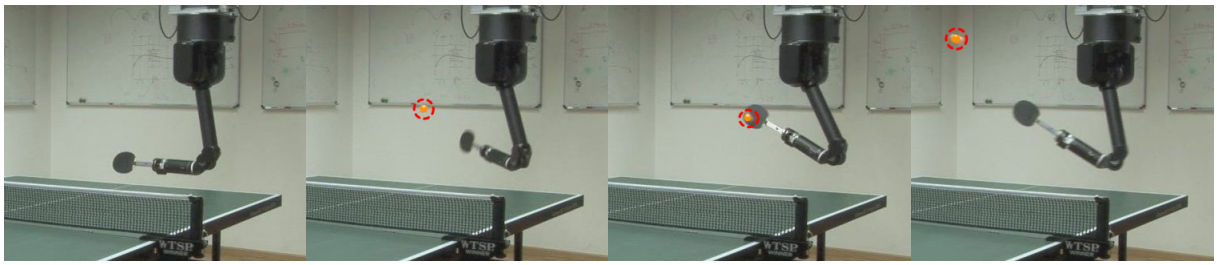
3.2 Computing the Meta-Parameters

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 point in time. Altogether, 15 parameters need to be determined, which include the timing parameter that determines the initiation of the hitting 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.

We can learn the 15 task parameters using CrKR as described in Section 2.2 based on the position and velocity of the ball above the net, which is used as the state of the system. The Euclidean distance of the ball and the racket at the estimated hitting time is used as the cost function. The accuracy of the system over time is shown in Figure 6. A more detailed description and evaluation of this approach 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

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



(b) Reproduced hitting motion by imitation learning.

Figure 7: 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.

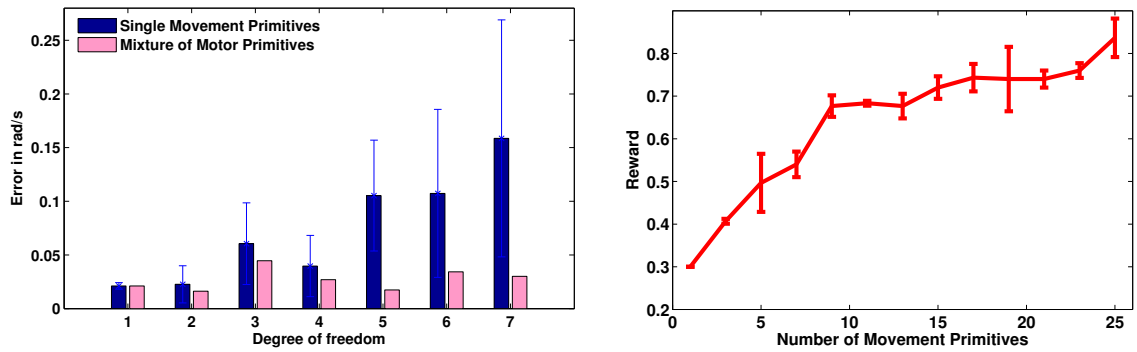
(Muelling et al., 2011).

3.3 MoMP

To analyze 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 physical 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.

3.3.1 Evaluation in Simulation

In order 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 et al., 2011). For the simulation of the ball, we neglected air drag in this evaluation. The system received the true position and velocity of the ball which reduces the sources of potential errors. 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 as described in Algorithm 3. 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.



(a) Execution Error

(b) Impovement Curve of MoMP

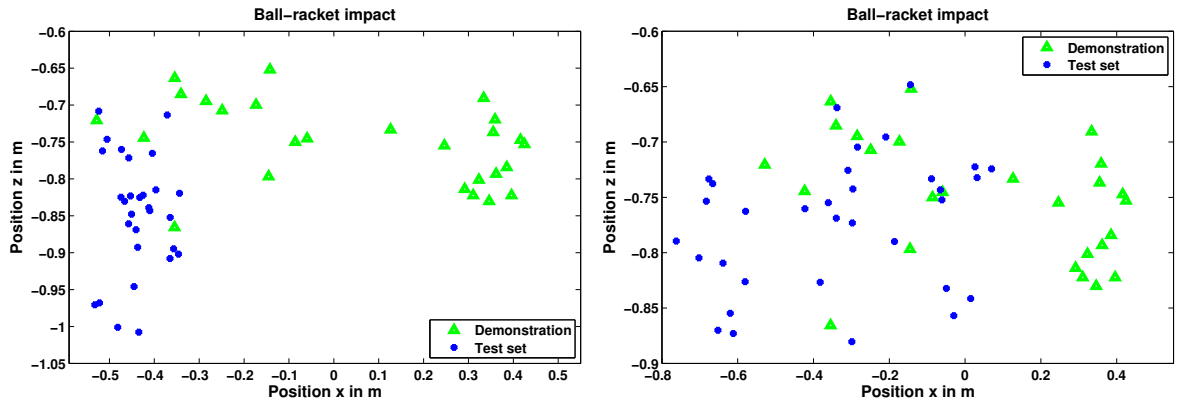
Figure 8: (a) Velocity error at the movement goal of the single movement primitives and the MoMP. The execution error of each movement primitive in the library was determined. The first bar of each DoF shows the mean error of the individual movement primitives and the corresponding standard deviations. The second bar of each DoF shows the mean error of the motor policies generated by the MoMP. (b) Improvement of the MoMP system with a growing number of movement primitives in the library.

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 opponents court. The combination of these components in an untrained mixture of movement primitives 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 a 94 % success rate. Analyzing the weight parameters for all movement primitives allowed the learning system to reduce the size of the library as all primitives π_i where γ_i converged to zero were effectively removed. We observed that about 27 % of the primitives were removed and that these primitives had an average performance of only a 30 % success rate.

3.3.2 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 7). 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. To ensure the safety of the teacher, reduced limits for velocity and acceleration were enforced. A second person manned the emergency shut-off. 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 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. First, we evaluated the performance of the individual movement primitives in our library. Analyzing the ability of each DMP to generalize to new situations (i.e., the ability to perform equally well in new situations), we found that not all DMPs were able to adapt to the changing velocity profiles as required. In the last three DoFs towards the racket, we found mean errors ranging from 0.02 rad/s to 0.4 rad/s from the desired



(a) Fixed ball launcher.

(b) Oscillating ball launcher.

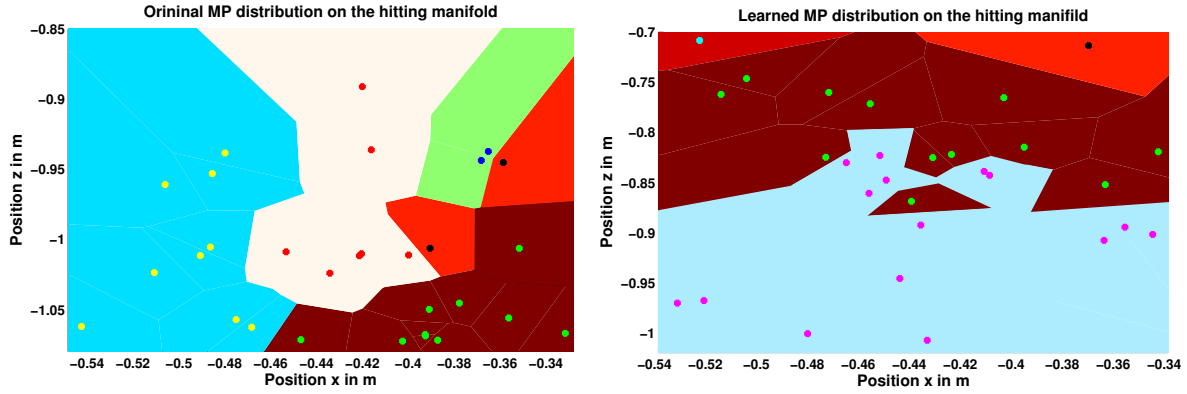
Figure 9: The locations of ball-racket impacts during demonstration and evaluation on the real robot. (a) The ball-racket impacts during the evaluation with the fixed ball launcher. (b) The ball-racket impacts of the evaluation of the oscillating ball launcher.

joint velocities. Using our MoMP system reduced the mean errors to 0.04 rad/s (see Figure 8(a)). Using just the average performance of the last three DoF as a reward signal, the system was able to reduce the average error to 0.02 rad/s where 8 movement primitives were mixed in average. The average performance of the system using MoMP without updating the gating network was 0.08 rad/s. Updating the gating network and 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 8(b).

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 three experiments. In the first two experiments we used the ball launcher to serve the ball to the robot to ensure similar conditions for both initial and final test phases. In the first experiment, the ball launcher was in an oscillating mode. The area served by the ball launcher measured 0.7 m x 0.4 m. The striking movement would be considered to be successful, if the ball was returned to the opponent's court. The setup was tested with 30 trials. Before training, the performance of the system was 69%. The distance between the bouncing points on the opponent's court and the desired target on the table had an average distance of 1.9 m to the target on the table. After training the system in 150 trials, the system returned 97% of the balls successfully to the opponent's court. The mean distance between the bouncing point and the target on the table was 0.6 m. Evaluating only the successful trials, the mean distance to the target was 0.46 m. The point of impact of the ball and the racket during demonstration and evaluation is shown in Figure 9.

In the second experiment, 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 for 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 10).

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 human opponent. Initially the robot was able to return 74.4 % of the balls. After playing one hour against the human, the robot was



(a) Before training.

(b) After training.

Figure 10: Distribution on the hitting manifold (i.e., the part of the state-space that defines the ball-racket contacts) of the most dominant movement primitives before (a) and after (b) training. Each colored region corresponds to one movement primitive in the movement library. Each point corresponds to the point of impact during evaluation. The usage of the movement primitives changed after training the system with a ball launcher. While two movement primitives were relocated, three were avoided and replaced by two better suited primitives. Please note that we have displayed only a small excerpt of the served forehand area here in order to visualize the re-organization of a few primitives.

	min $\dot{x}$	max $\dot{x}$	min $\dot{y}$	max $\dot{y}$	min $\dot{z}$	max $\dot{z}$	$v_{\max}$
Simulation	-0.92 m/s	0.34 m/s	1.88 m/s	4.03 m/s	-2.11 m/s	1.23 m/s	4.64 m/s
Experiment 1	-0.66 m/s	0.00 m/s	2.8 m/s	3.27 m/s	-2.75 m/s	-1.03 m/s	4.33 m/s
Experiment 2	-0.49 m/s	-0.22 m/s	2.59 m/s	2.99 m/s	-3.04 m/s	-1.55 m/s	4.29 m/s
Experiment 3	-0.92 m/s	0.57 m/s	2.03 m/s	4.04 m/s	-2.02 m/s	0.29 m/s	4.61 m/s

Table 1: Variance of the velocity of the ball before impact in the different experiments of the evaluation. The maximal velocity $v_{\max}$ of the ball before impact is defined by $v_{\max} = \sqrt{|\dot{x}|^2 + |\dot{y}|^2 + |\dot{z}|^2}$. All values are given in meters per second [m/s].

able to return 88 % of the balls successfully, i.e., to the opponent's court.

The information about the velocity of the ball before impact, as well as on the variance of the incoming ball's direction are summarized for all experiments in Table 1. The mean error of the desired final position and velocity at time T_f during evaluation on the real robot was 0.0017 rad and 0.0431 rad/s, respectively. A systematic study of the errors in simulation showed that changes in the position affects the final error more than changes in the velocity. The results from this analysis are depicted in Table 2.

4 Conclusion

In this paper, we presented a new framework that enables a robot to learn basic cooperative table tennis from demonstration and interaction with a human player. 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 the proposed 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. Our experiments showed that both (i) selecting movement primitives from a library based on the current task

Setup	Goal Perturbation				Resulting Error			
	δg_f [rad]		$\delta \dot{g}_f$ [rad/s]		e_{g_f} [rad]		$e_{\dot{g}_f}$ [rad/s]	
	average	**max**	**average**	**max**	**mean**	**std**	**mean**	**std**
real robot evaluation	0.44	1.93	1.54	3.01	0.0017	0.0013	0.0431	0.0375
	min	**max**	**min**	**max**				
simulation analysis (5 configurations)	−2.00	2.00	0.00	0.00	0.0054	0.0056	0.1695	0.2323
	−1.00	1.00	0.00	0.00	0.0035	0.0030	0.0929	0.1083
	0.00	0.00	−2.00	2.00	0.0031	0.0024	0.0597	0.0701
	0.00	0.00	−1.00	1.00	0.0028	0.0024	0.0566	0.0706
	−1.00	1.00	−2.00	2.00	0.0036	0.0031	0.0950	0.0944

Table 2: Analysis of the accuracy of the modified hitting primitives after both imitation and reinforcement learning. The right part of the table shows the changes in the desired final position and velocity compared to the values in the demonstration. On the left side of the table the corresponding mean errors (averaged over all DoFs) in the desired position and velocity are displayed at time T_f. The simulation analysis consisted of several configurations in which the motor policies yield through demonstrations were perturbed. For each configuration, either the final position, final velocity or both were modified using uniform value shifts between [-2,2] and [-1,1]. Please note that these position shifts exceed those which were observed on the real system.

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 return balls served to the forehand area of the robot successfully to the opponent's court. We managed to perform and improve the movements involved in table tennis with a reasonable amount of training data for real robot evaluation.

The presented method of generating motor policies from a library of demonstrated movements is not limited to the demonstrated show-case of robot table tennis. In particular, the approach is suited for many tasks where a set of movements has to be adapted to new, different but similar situations as commonly found in striking sports. Additionally, it could potentially be adapted to other goal-oriented movements (e.g., simple manipulation tasks, foot-step generation for walking over rough terrain, etc.) where our approach would be a good starting point.

Our approach generalizes well if the situations are within the convex combination of demonstrations. For extrapolating, more additional reinforcement learning will be needed as exploration is required. In our approach, only local reinforcement learning is employed for self improvement. As a result, the solution can be extrapolated to new similar situations and movements but not totally novel ones. Discovering a movement that differs from all existing ones (as e.g., Dick Fosbury did in high-jumping with the Fosbury flop (Wikipedia, 2012)) is clearly a limit of this data-driven framework.

Preliminary results of this work were presented in Muelling et al. (2010) and Muelling et al. (2012). In future work, we will include opponent prediction (Wang et al., 2012) in order to infer the hitting area from the hitting movement of the opponent. As a result, the system has more time to switch safely between fore- and backhand. Thus, the system will be able to safely return incoming balls on the whole width of the table.

5 Acknowledgments

This research was supported by the European Community's Seventh Framework Programme under grant agreement no. ICT-270327 CompLACS.

References

Acosta, L., Rodrigo, J., Mendez, J., Marchial, G., and Sigut, M. (2003). Ping-pong player prototype. *IEEE Robotics and Automation Magazine*, 10(4):44–52.

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

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*, pages 1371–1394. Springer.

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

Bishop, C. and Tipping, M. (2003). *Bayesian Regression and Classification*, volume 190, pages 267 – 289. IOS Press.

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.

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

Fässler, H., Beyer, H. A., and Wen, J. T. (1990). A robot ping pong player: optimized mechanics, high performance 3d vision, and intelligent sensor control. *Robotersysteme*, 6(3):161–170.

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.

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).

Hartley, J. (1987). Toshiba progress towards sensory control in real time. *The Industrial Robot*, 14(1):50–52.

Hashimoto, H., Ozaki, F., Asano, K., and Osuka, K. (1987). Development of a ping pong robot system using 7 degrees of freedom direct drive. In *Industrial Applications of Robotics and Machine Vision (IECON)*, pages 608–615.

Huang, Y., Xu, D., Tan, M., and Su, H. (2011). Trajectory prediction of spinning ball for ping-pong robot. In *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3434 – 3439.

Ijspeert, A. J., Nakanishi, J., and Schaal, S. (2002). Movement imitation with nonlinear dynamical systems in humanoid robots. In *IEEE International Conference on Robotics and Automation (ICRA)*.

International Table Tennis Federation (2011). Table tennis rules. http://www.ittf.com/ittf_handbook/hb.asp?s_number=2.

Jacobs, R., Jordan, M., Nowlan, S., and Hilton, G. (1991). Adaptive mixures of local experts. *Neural Computation*, 3:79 – 87.

Knight, J. and Lowery, D. (1986). Pingpong-playing robot controlled by a microcomputer. *Microprocessors and Microsystems*, 10(6):332–335.

Kober, J., Mohler, B., and Peters, J. (2008). Learning perceptual coupling for motor primitives. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*.

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

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

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

Kroemer, O., Detry, R., Piater, J., and Peters, J. (2010). Grasping with vision descriptors and motor primitives. In *Proceedings of the International Conference on Informatics in Control, Automation and Robotics (ICINCO)*.

Lai, C. and Tsay, J. (2011). Self-learning for a humanoid robot ping-pong player. *Advanced Robotics*, 25:1183 – 1208.

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. *Neural Networks*, 9:1281–1302.

Miyazaki, F., Matsushima, M., and Takeuchi, M. (2005). Learning to dynamically manipulate: A table tennis robot controls a ball and rallies with a human being. In *Advances in Robot Control*, pages 3137–341. Springer.

Muelling, K., Kober, J., Kroemer, O., and Peters, J. (2012). Learning to select and generalize striking movements in robot table tennis. In *Proceedings of the AAAI 2012 Fall Symposium on robots that Learn Interactively from Human Teachers*.

Muelling, K., Kober, J., and Peters, J. (2010). Learning table tennis with a mixture of motor primitives. In *10th IEEE-RAS International Conference on Humanoid Robots (HUMANOIDS)*.

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.

Ning, K., Kulvicius, T., Tamosiunaite, M., and Wörgötter, F. (2011). Accurate position and velocity control for trajectories based on dynamic movement primitives. In *IEEE International Conference on Robotics and Automation (ICRA), Shanghai, China, 9-13 May 2011*, pages 5006–5011.

Park, D., Hoffmann, H., Pastor, P., and Schaal, S. (2008). Movement reproduction and obstacle avoidance with dynamic movement primitives and potential fields. In *IEEE International Conference on Humanoid Robots (Humanoids)*, pages 91 –98.

Peters, J. and Schaal, S. (2006). Policy gradient methods for robotics. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*.

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

Peters, J. and Schaal, S. (2008b). Reinforcement learning of motor skills with policy gradients. *Neural Networks*, 21(4):682–697.

Pongas, D., Billard, A., and Schaal, S. (2005). Rapid synchronization and accurate phase-locking of rhythmic motor primitives. In *IEEE/RSJ International Conference Intelligent Robots and Systems*, pages 2911–2916.

Rasmussen, C. and Williams, C. (2006). *Gaussian Processes for Machine Learning*. MIT Press, Cambridge, MA, USA.

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

Schaal, S., Atkeson, C. G., and Vijayakumar, S. (2002). Scalable techniques from nonparameteric statistics for real-time robot learning. *Applied Intelligence*, (1):49 – 60.

Schaal, S., Ijspeert, A., and Billard, A. (2003a). 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., Peters, J., Nakanishi, J., and Ijspeert, A. (2003b). Learning motor primitives. In *International Symposium on Robotics Research (ISRR)*.

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.

Wang, Z., Deisenroth, M., Amor, H., Vogt, D., Schoelkopf, B., and Peters, J. (2012). Probabilistic modeling of human movements for intention inference. In *Proceedings of Robotics: Science and Systems (R:SS)*.

Wikipedia (2012). Fosbury Flop. http://en.wikipedia.org/wiki/Fosbury_Flop.

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)*, pages 1609–1616.

Zhang, Z. an Xu, D. and Tau, M. (2010). Visual measurements and prediction of ball trajectory for table tennis robot. *IEEE Transactions on Instrumentation and Measurements*, 59(12):3195–3205.

Appendix

Appendix A - Index to Multimedia Extensions

Appendix B - Glossary

To indicate matrices, we use bold upper case letters. For vectors, we use bold lower case letters. Italic letters indicate scalars.

Extension	Media Type	Description
1	Video	The video visualizes the different steps in adapting the movement generation based on the outcome of previous trials. We also provided the video online: `www.youtube.com/watch?v=SH3bADiB7uQ`

movement primitive	A sequence of motor commands executed to accomplish a given motor task.
state of the system $\mathbf{s}$	Vector of variables necessary to describe the system. In the table tennis task the state of the system refers to the current position and velocity of the ball.
motor skill	Learned sequence of smooth movements executed in order to master a particular task.
movement library	Database of movement primitives stored as motor policies.
motor policy $\pi(\mathbf{x})$	Function that maps the state $\mathbf{s}$ of a movement system and the internal parameters $\mathbf{w}$ to a control vector that defines the executable motion. Here, we use the augmented state $\mathbf{x}$ instead of the state of the system $\mathbf{s}$.
meta-parameters $\boldsymbol{\delta}$	The movement goal, duration and amplitude of a movement described by a motor policy.
augmented state $\mathbf{x}$	Parameter vector consisting of the state $\mathbf{s}$ and a set of parameters that describes the characteristics of the current task.
movement goal	The finite state of the desired movement.
hitting manifold	The part of state-space that defines the ball-racket contacts.

Appendix C - Cost regularized Kernel Regression

Cost-regularized Kernel Regression (CrKR), as suggested in (Kober et al., 2012), is a reinforcement learning approach based on a kernelized version of reward-weighted regression. The goal is to find a mapping between the meta-parameter $\boldsymbol{\delta}$ of a motor policy and a situation described by the state vector $\mathbf{s}$. Therefore, we looking for a stochastic policy $\mu(\boldsymbol{\delta}|\mathbf{s})$ that maximizes the expected reward

$$J(\mu) = \mathbb{E}\left\{r(\mathbf{s},\boldsymbol{\delta})|\mu\right\}, \tag{13}$$

where $r(\mathbf{s},\boldsymbol{\delta})$ denotes the reward following the selection of $\boldsymbol{\delta}$ in state $\mathbf{s}$. The CrKR is based on the use of the Gaussian distribution with mean $\bar{\boldsymbol{\delta}}$ and variance $\boldsymbol{\Sigma}$ for the policy μ, i.e., $\mu(\boldsymbol{\delta}|\mathbf{s}) = \mathcal{N}\left(\boldsymbol{\delta}|\bar{\boldsymbol{\delta}},\boldsymbol{\Sigma}\right)$. The mean and the variance are computed by

$$\bar{\boldsymbol{\delta}} = \mathbf{k}(\mathbf{s})^T(\mathbf{K}+\lambda\mathbf{C})^{-1}\mathbf{D} \tag{14}$$

$$\boldsymbol{\Sigma} = k(\mathbf{s},\mathbf{s}) + \lambda - \mathbf{k}(\mathbf{s})^T(\mathbf{K}+\lambda\mathbf{C})^{-1}\mathbf{k}(\mathbf{s}) \tag{15}$$

where $\mathbf{D}$ is a matrix that contains the training examples $\boldsymbol{\delta}_n^T$, $\mathbf{C} = \mathrm{diag}\{\mathrm{r}_1^{-1}(\mathbf{x}_1,\mathbf{s}_1),...,\mathrm{r}_\mathrm{N}^{-1}(\mathbf{x}_\mathrm{N},\mathbf{s}_\mathrm{N})\}$ is the cost matrix, λ is a ridge factor, $\mathbf{k}(\mathbf{s})$ is a vector that measures the distance between the current state and the state of the training example using a Gaussian kernel and $\mathbf{K}$ is a matrix that contains the pairwise distances between all combinations of training points. A detailed derivation of this policy can be found in (Kober et al., 2012).

In this policy μ, the variance corresponds to the uncertainty over output $\boldsymbol{\delta}$ and a high cost results in a high uncertainty. If we have a training example with a high cost, the policy will explore new actions and the training point will only have a strongly reduced influence. On the other hand, when the cost is low, we can assume that we are close to the optimal policy and the variance shrinks to a small region around the observed example.

Appendix D - Linear Bayesian Regression

Linear Bayesian Regression is a linear regression approach in the context of Bayesian inference. The following summary of the approach is based on Bishop (2006). For more details we refer the reader to

Chapter 2 of Rasmussen and Williams (2006) or Chapter 3 of Bishop (2006).

Assume we are interested in the target variable t which can be modeled by a function $y(\mathbf{x}, \boldsymbol{\theta})$ with additive Gaussian noise such that

$$\begin{aligned} t &= y(\mathbf{x}, \boldsymbol{\theta}) + \epsilon \\ &= \boldsymbol{\theta}^T \phi(\mathbf{x}) + \epsilon, \end{aligned}$$

where $\phi(\mathbf{x})$ is the vector of basis functions, $\boldsymbol{\theta}$ is the vector of weights and ϵ is a zero mean Gaussian random variable with precision β. The conditional distribution of the target variable $\mathbf{t}$ given $\mathbf{x}$, $\boldsymbol{\theta}$ and β is given by the Gaussian distribution $p(t|\mathbf{x}, \boldsymbol{\theta}, \beta) = \mathcal{N}(t|\boldsymbol{\theta}^T \boldsymbol{\phi}(\mathbf{x}), \beta^{-1})$ (Bishop and Tipping, 2003). Using the conjugate prior $p(\boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{\theta}|\mathbf{m}_0, \mathbf{V}_0)$, the posterior distribution of the unknown parameter vector $\boldsymbol{\theta}$ is again a Gaussian distribution and given by $p(\boldsymbol{\theta}|\mathbf{t}) = \mathcal{N}\boldsymbol{\theta}|\mathbf{m}_n, \mathbf{V}_N)$, where

$$\begin{aligned} \mathbf{V}_N &= \left(\mathbf{V}_0^{-1} + \beta \boldsymbol{\phi}^T \boldsymbol{\phi}\right)^{-1}, \\ \mathbf{m}_N &= \mathbf{V}_N \left(\mathbf{V}_0^{-1} \mathbf{m}_0 + \beta \boldsymbol{\Phi}^T \mathbf{t}\right), \end{aligned}$$

and $\mathbf{m}_0$ and $\mathbf{V}_0$ are the mean and covariance of the prior of $\boldsymbol{\theta}$ (Bishop, 2006).

We can weight each observation similarly to a weighted least squares regression by assigning bad observations a higher variance than others. Therefore, we obtain $p(t|\mathbf{x}, \boldsymbol{\theta}, \beta) = \mathcal{N}(t|\boldsymbol{\theta}^T \boldsymbol{\phi}(\mathbf{x}), (\beta w)^{-1})$. Following the derivation in Bishop (2006), mean and covariance of the posterior distribution of $\boldsymbol{\theta}$ are defined by

$$\begin{aligned} \mathbf{V}_N &= \left(\mathbf{V}_0^{-1} + \beta \boldsymbol{\phi}^T \mathbf{W} \boldsymbol{\phi}\right)^{-1}, \\ \mathbf{m}_N &= \mathbf{V}_N \left(\mathbf{V}_0^{-1} \mathbf{m}_0 + \beta \boldsymbol{\Phi}^T \mathbf{W} \mathbf{t}\right), \end{aligned}$$

where $\mathbf{W}$ is a diagonal matrix whose diagonal entries define the uncertainty of the observation. When considering the special case of a Gaussian prior with zero mean $p(\boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{\theta}|\mathbf{0}, \alpha^{-1}\mathbf{I})$ (Bishop, 2006), the corresponding mean and covariance of the posterior distribution are given by

$$\begin{aligned} \mathbf{V}_N &= \left(\alpha \mathbf{I} + \beta \boldsymbol{\Phi}^T \mathbf{W} \boldsymbol{\Phi}\right)^{-1}, \\ \mathbf{m}_N &= \beta \mathbf{V}_N \boldsymbol{\Phi}^T \mathbf{W} \mathbf{t}. \end{aligned}$$

In this case, the mean $\mathbf{m}_N$ corresponds to the solution of the Weighted Ridge Regression.

Appendix E - Stability of Hitting DMPs

Here, we provide the mathematical proof of the stability and convergence of the extension of the DMPs presented in Kober et al. (2010) and in this paper. Please note that for the ease of reading the variables used in this section are not related to those with the same name elsewhere in this manuscript. The block diagrams for the second-order-system for both systems is illustrated in Figure 11.

Stability of Hitting DMPs with Linear Velocity

If we assume a linear velocity of the target object as in the setup of Kober et al. (2010), the forward channel of the second-order-system expressed in Laplace space is given by

$$G(s) = \frac{Y(s)}{E(s)},$$

where $Y(s) = (AE(s) + BsE(s))/s^2$ is the state of our system, $U(s)$ is our desired position of the moving target, $E(s) = U(s) - Y(s)$ and $s \in \mathbb{C}$ is the (complex) frequency variable. The transfer function is given

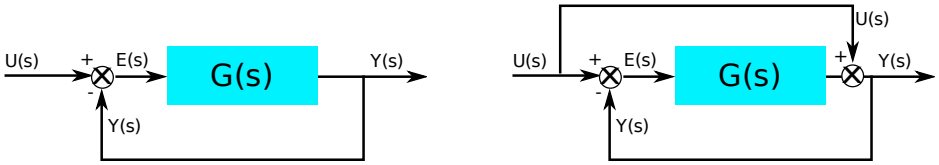


Figure 11: Diagrams of the employed second-order-system for the (a) DMPs with linear velocity and (b) the DMPs with polynomial velocity in Laplace space.

by

$$T(s) = \frac{Y(s)}{U(s)} = \frac{G(s)}{1 + G(s)} = \frac{A + Bs}{s^2 + Bs + A}. \tag{16}$$

Using $A = \omega^2$ and $B = 2\omega$ the system has a double pole at $-\omega$ which results in a critically damped response. The steady-state error can be calculated by

$$e_{ss} = \lim_{t \to \infty} e(t) = \lim_{s \to 0} sU(s)(1 - T(s)). \tag{17}$$

For a linear moving object $U(s) = V/s^2 + C/s$ and therefore we obtain

$$e_{ss} = 0. \tag{18}$$

The error's change rate is given by

$$\dot{e}_{ss} = 0. \tag{19}$$

Stability of Hitting DMPs with a Polynomial Velocity Profile

For the extension of the DMPs used and presented in this paper, the feed-forward term $\tau^2 \ddot{g}$ cancels all higher order terms, hence the hitting DMPs with polynomial velocity profile has the same stability properties as the hitting DMPs with linear velocity.

Appendix F - Review of Robot Table Tennis

Driven by a robot-table tennis competition series, several early systems were presented between 1983 and 1993 (Andersson, 1988; Knight and Lowery, 1986; Hartley, 1987; Hashimoto et al., 1987; Fässler et al., 1990). The focus in that period was mainly on the design of efficient vision systems and improved hardware rather than finding robust algorithms to achieve a performance comparable to humans. An important breakthrough was achieved by Andersson (1988) who presented the first robot table tennis system which was capable to play against humans. Therefore, Andersson used a high speed vision system and a six DoF Puma 260 arm with a 0.45 m long stick mounted between table tennis racket and robot. After the competition ended in 1993 several groups picked up robot table tennis and explored certain aspects of the setup such as vision and trajectory prediction (Zhang and Tau, 2010; Huang et al., 2011), construction of low-cost systems (Acosta et al., 2003) or predicting the hitting point (Miyazaki et al., 2005; Lai and Tsay, 2011). Miyazaki et al. (2005) proposed a robot table tennis system mounted on the table with four DoF consisting of two linear axis and a two DoF pan-tilt unit. The group applied locally weighted regression to predict the impact time, ball position and velocity. Lai and Tsay (2011) proposed a setup to estimate the ball trajectory using fuzzy adaptive resonance theory network and learn the orientation angles of the racket using Self-organizing maps.

None of the approaches concentrated on learning robust hitting movements that are able to compensate for perturbations in the environment and inaccuracies in the prediction of the impact point and time.

Muelling et al. (2011) proposed a table tennis setup where the hitting motion is modeled using fifth order splines. The use of autonomous movement representations as DMPs enable us to learn the movement without a programmer specifying a fixed movement plan for different kind of movements. DMPs allow to represent arbitrary shaped movements for many different tasks. The shape of a movement can contain important features necessary to perform the task successfully. Furthermore, DMPs are robust against perturbations, as they do not directly depend on time. As a result, we are able to change the movement goal and timing online while retaining the shape of the movement.