Learning Motor Primitives for Robotics

Jens Kober, Jan Peters
Robot Learning Lab (RoLL), Department of Empirical Inference
Max Planck Institute for Biological Cybernetics
Spemannstr. 38, 72076 Tübingen, Germany
{jens.kober,jan.peters}@tuebingen.mpg.de

***Abstract*— The acquisition and self-improvement of novel motor skills is among the most important problems in robotics. Motor primitives offer one of the most promising frameworks for the application of machine learning techniques in this context. Employing an improved form of the dynamic systems motor primitives originally introduced by Ijspeert et al. [2], we show how both discrete and rhythmic tasks can be learned using a concerted approach of both imitation and reinforcement learning. For doing so, we present both learning algorithms and representations targeted for the practical application in robotics. Furthermore, we show that it is possible to include a start-up phase in rhythmic primitives. We show that two new motor skills, i.e., Ball-in-a-Cup and Ball-Paddling, can be learned on a real Barrett WAM robot arm at a pace similar to human learning while achieving a significantly more reliable final performance.**

I. Introduction

To date, most robots are still programmed by a smart operator who uses human understanding of the desired task in order to create a program for accomplishing the required behavior. While such specialized programming is highly efficient, it is also expensive and limited to the situations the human operator considered. In order to create robots that can acquire new skills or improve existing abilities autonomously, learning is the key ability. However, off-the-shelf machine learning techniques do not scale into the high-dimensional domains of anthropomorphic robotics. Instead, robot learning requires methods that employ both representations and algorithms appropriate for the domain. When humans learn new motor skills, e.g., paddling a ball with a table-tennis racket or hitting a tennis ball, it is highly likely that they rely on a small set of motor primitives and use imitation as well as reinforcement learning [1]. Inspired by this example, we will discuss the technical counterparts in this paper and show how both single-stroke and rhythmic tasks can be learned efficiently by mimicking the human presenter with subsequent reward-driven self-improvement.

Recently, the idea of using dynamical systems as motor primitives was put forward by Ijspeert et al. [2] as a general approach for representing control policies for basic movements. The resulting movement generation has a variety of favorable properties, i.e., rescalability with respect to both time and amplitude, basic stability properties and the possibility to encode either single-stroke or rhythmic behaviors. If a favorable function approximator is chosen in this context, ideally one that is linear in its parameters, then learning can be sufficiently fast for application in robotics in real-time. A series of such frameworks has been introduced to date [2]–[5]. Previous applications include a variety of different basic motor skills such as tennis swings [2], T-ball batting [6], drumming [7], planar biped walking [3], [8], constrained reaching tasks [9] and even in tasks with potential industrial application [10]. Nevertheless, most of the previous work in motor primitive learning (with the exceptions of [6] and [9]) has focused on learning by imitation *without* subsequent self-improvement. In real life, a human demonstration is rarely ever perfect nor does it suffice for near-optimal performance. Thus, additional reinforcement learning is essential for both performance-based refinement and continuous adaptation of the presented skill. Note, that this approach is substantially different from the complementary idea of apprenticeship learning [11] that attempts to infer the intent of the teacher and learn his policy through an inverse reinforcement learning approach.

In this paper, we present our current best performing setups for motor primitive learning with both the required methods for imitation and reinforcement learning. The motor primitive framework is discussed both for discrete, single-stroke movement as well as rhythmic movements in Section II. The appropriate imitation and reinforcement learning methods are given in Section III. In Section IV, we show how the resulting framework can be applied to both learning *Ball-in-a-Cup* as a discrete task and *Ball-Paddling* as a rhythmic task on a real Barrett WAM. The accompanying video[1] shows human presentation and imitation for both tasks as well as reinforcement learning for Ball-in-a-Cup. The ball-paddling task is of particular interest as we show how the combination of different motor primitives is possible. It is among the first applications where both rhythmic and discrete dynamic systems motor primitives [2] are used in conjunction to achieve the task.

Note that the presented work on Ball-in-a-Cup differs from our previous work in [5] where we employed an extended formulation of the motor primitives in simulation while we use the standard formulation here and have results on a physical Barrett WAM. In [12], we give more details on the derivation of the PoWER algorithm while the robot implementation is not treated in detail. Our work on Ball-Paddling with rhythmic motor primitives is recent work where we cannot offer any additional background reading.

[1] On-line at http://www.YouTube.com/watch?v=cNYoMVZQdYM.

II. Motor Primitive Representations

In this section, we first introduce the general idea behind dynamic system motor primitives as suggested in [2]. Subsequently, we discuss the details of both the most current version of discrete and the rhythmic motor primitives based on [4].

A. Generic Idea

The generic idea of dynamic systems motor primitives as suggested in [2] can be understood in the most simple form as two dynamical system with a one-way connection such that one system drives the other one. Here, the hidden system acts as an adjustable clock or phase of the movement. As a result, the dynamical systems motor primitives can be parted into two components, i.e., a canonical system h which drives transformed systems $\mathbf{g}_k$ for every considered degree of freedom k; system h couples and synchronizes the different degrees of freedom. For combining motor primitives one would have seperate canonical systems h if they are independent or a joint one if they are dependent. As a result, we have a system of differential equations given by

$$\dot{z} = h(z), \tag{1}$$
$$\dot{\mathbf{x}} = \mathbf{g}(\mathbf{x}, z, \mathbf{w}), \tag{2}$$

which determine the variables of internal focus $\mathbf{x}$. Here, z denotes the state of the canonical system and $\mathbf{w}$ the internal parameters for transforming the output of the canonical system. The output of the transformed systems are desired positions, velocities and accelerations either in joint space or in task space. A suitable controller is used to convert these in motor torques.

The original formulation in [2] has the advantage that the choice of the dynamical systems in Equations (1, 2) allows determining the stability of the movement, choosing between a rhythmic and a discrete movement and is invariant under rescaling in both time and movement amplitude. With the right choice of function approximator (in our case locally linear, see Section III-A), fast learning from a teacher's presentation is possible. Additional feedback terms can be added as in [2], [4], [5], [13].

B. Discrete Movement Primitives

While the original formulation in [2] for discrete dynamical systems motor primitives used a second-order canonical system, this formulation has proven to be unnecessarily complicated in practice. Since then, it has been simplified and it could be shown that a single first order system suffices [4]

$$\dot{z} = h(z) = -\tau \alpha_h z, \tag{3}$$

which represents the phase of the movement. It has a time constant τ (the same as in the transformed system) and a parameter α_h which is chosen such that the system is stable. We can now choose our internal state of the transformed system such that position of degree of freedom k is given by $q_k = x_{2k}$, i.e., the $2k^{\text{th}}$ component of $\mathbf{x}$, the velocity by $\dot{q}_k = \tau x_{2k+1} = \dot{x}_{2k}$ and the acceleration by $\ddot{q}_k = \tau \dot{x}_{2k+1}$. Upon these assumptions, we can express the motor primitives function $\mathbf{g}$ in the following form

$$\dot{x}_{2k+1} = \tau \alpha_g (\beta_g (t_k - x_{2k}) - x_{2k+1}) + \tau \left((t_k - x_{2k}^0) + a_k \right) f_k, \tag{4}$$
$$\dot{x}_{2k} = \tau x_{2k+1}, \tag{5}$$

with k denoting the degree of freedom. This differential equation has the same time constant τ as the canonical system, appropriately set parameters α_g, β_g, a goal parameter t_k, an amplitude modifier a_k, the inital position x_{2k}^0, and a transformation function f_k. This function alters the output of the canonical system so that the transformed system can represent complex nonlinear patterns and it is given by

$$f_k(z) = \textstyle\sum_{i=1}^{N} \psi_i(z)\, w_i z \tag{6}$$

where $\mathbf{w}$ are adjustable parameters and $\boldsymbol{\psi}(z)$ are weights [4]. It uses normalized Gaussian kernels as weights given by

$$\psi_i = \frac{\exp\left(-h_i (z - c_i)^2\right)}{\sum_{j=1}^{N} \exp\left(-h_j (z - c_j)^2\right)}. \tag{7}$$

These weights localize the interaction in phase space using the centers c_i and widths h_i.

C. Rhythmic Movement Primitives

Similar as for the discrete dynamic systems motor primitives, the original formulation in [2] has been superceded by a newer version for which a single first order dynamical system suffices [4]. Thus, we have a canonical system $\dot{z} = h(z) = \tau\omega$ with ω as the phase rate of change and can be used for coupling several degrees of freedom together, see [3]. The motor primitives function $\mathbf{g}$ can be given in the form

$$\dot{x}_{2k+1} = \tau \alpha_g (\beta_g (x_m - x_{2k}) - x_{2k+1}) + \tau a_k f_k, \tag{8}$$
$$\dot{x}_{2k} = \tau x_{2k+1}. \tag{9}$$

This differential equation has the same time constant τ as the canonical system, appropriately set parameters α_g, β_g, the baseline of the oscillation x_m, an amplitude modifier a_k, and a transformation function f_k. The transformation function transforms the output of the canonical system so that the transformed system can represent complex nonlinear patterns and it is given by $f_k(z) = \sum_{i=1}^{N} \psi_i(z)\, w_i$ where $\mathbf{w}$ are adjustable parameters and $\boldsymbol{\psi}(z)$ are weights. It uses normalized Gaussian kernels as weights given by

$$\psi_i = \frac{\exp\left(-h_i (1 - \cos(z - c_i))\right)}{\sum_{j=1}^{N} \exp\left(-h_j (1 - \cos(z - c_j))\right)}, \tag{10}$$

where the main difference to Equation (7) is cosine transformation of the localization [4]. Again, the weights localize the interaction in phase space using the centers c_i and widths h_i.

III. Learning Methods for Motor Primitives

It is likely that humans rely both on imitation and on reinforcement learning for learning new motor skills as both of these approaches have different functions in the learning process. Imitation learning has a given target and, thus, it allows to learn policies from the examples of a teacher. However, imitation learning can only reproduce a policy representing or generalizing an exhibited behavior. Self-improvement by trial-and-error with respect to an external reward signal can be achieved by reinforcement learning. Nevertheless, traditional reinforcement learning algorithms require exhaustive exploration of the state and action space. Given the high-dimensionality of the state-space of anthropomorphic robots (a seven degree of freedom robot defies exhaustive exploration), the "curse of dimensionality" [14] fully applies and we need to rely on local reinforcement learning methods which improve upon the preceding imitation instead of traditional 'brute force' approaches. To some extent, this mimicks how children acquire new motor skills with the teacher giving a demonstration while the child subsequently attempts to reproduce and improve the skill by trial-and-error. However, note that not every task requires reinforcement learning and some can be learned purely based on imitations. Nevertheless, few tasks are known which are directly learned by reinforcement learning without preceding mimicking [4]. Thus, we first review how to do imitation learning with dynamic systems motor primitives in Section III-A and, subsequently, we show how reinforcement learning can be applied in this context in Section III-B. The latter section will outline our reinforcement learning algorithm for the application in motor primitive learning.

A. Imitation Learning for Discrete & Rhythmic Dynamical Motor Primitives

In the presented framework, we initialize the motor primitives by imitation learning as in [4]. This step can be performed efficiently in the context of dynamical systems motor primitives as the transformation function (6) is linear in parameters. As a result, we can choose the weighted squared error

$$\varepsilon_m^2 = \textstyle\sum_{i=1}^n \psi_i^m \left(f_i^{\text{ref}} - \mathbf{z}_i^{\mathrm{T}} \mathbf{w}^m\right)^2 \tag{11}$$

as cost function and minimize it for all parameter vectors $\mathbf{w}^m$ with $m \in \{1, 2, \ldots, M\}$. Here, the corresponding weighting function are denoted by ψ_i^m and the basis functions by $\mathbf{z}_i^{\mathrm{T}}$. The reference or target signal f_i^{ref} is the desired transformation function and $i \in \{1, 2, \ldots, n\}$ indicates the number of the sample. The error in Equation (11) can be rewritten as

$$\varepsilon_m^2 = \left(\mathbf{f}^{\text{ref}} - \mathbf{Z}\mathbf{w}^m\right)^{\mathrm{T}} \mathbf{\Psi} \left(\mathbf{f}^{\text{ref}} - \mathbf{Z}\mathbf{w}^m\right) \tag{12}$$

with $\mathbf{f}^{\text{ref}}$ giving the value of f_i^{ref} for all samples i, $\mathbf{\Psi} = \operatorname{diag}\left(\psi_i^m, \ldots, \psi_n^m\right)$ and $\mathbf{Z}_i = \mathbf{z}_i^{\mathrm{T}}$. As a result, we have a standard locally-weighted linear regression problem that can be solved straightforwardly and yields the unbiased estimator

$$\mathbf{w}^m = \left(\mathbf{Z}^{\mathrm{T}} \mathbf{\Psi} \mathbf{Z}\right)^{-1} \mathbf{Z}^{\mathrm{T}} \mathbf{\Psi} \mathbf{f}^{\text{ref}}. \tag{13}$$

This general approach has originally been suggested in [2]. Estimating the parameters of the dynamical system is slightly more daunting, i.e., the movement duration of discrete movements is extracted using motion detection and the time-constant is set accordingly. Similarly, the base period for the rhythmic dynamical motor primitives was extracted using first repetitions and, again, the time-constants τ are set accordingly. As the start-up phase in rhythmic presentations may deviate significantly from the periodic movement, the baseline of the oscillation $\mathbf{x}_m$ often needs to be estimated based on the later part of the recorded movement, the value $\mathbf{a}$ is determined as the mean of the amplitudes of individual oscillations in this part.

B. Reinforcement Learning with PoWER

Reinforcement learning [15] of motor primitives is a very specific type of learning problem where it is hard to apply generic reinforcement learning algorithms [6], [16]. For this reason, the focus of this paper is largely on novel domain-appropriate reinforcement learning algorithms which operate on parametrized policies for episodic control problems.

1) Reinforcement Learning Setup: When modeling our problem as a reinforcement learning problem, we always have a high-dimensional state $\mathbf{s} = [\mathbf{z}, \mathbf{x}]$ and as a result, standard RL methods which discretize the state-space can no longer be applied. The action $\mathbf{a} = \mathbf{f}(\mathbf{z}) + \hat{\boldsymbol{\epsilon}}$ is the output of our motor primitives where the exploration is denoted by $\boldsymbol{\epsilon}$. As a result, we have a stochastic policy $\mathbf{a} \sim \pi(\mathbf{s})$ with parameters $\boldsymbol{\theta} = [\mathbf{w}^i] \in \mathbb{R}^N$ which can be seen as a distribution over the actions given the states. After a next time-step δt, the actor transfers to a state $\mathbf{s}_{t+1}$ and receives a reward r_t. As we are interested in learning complex motor tasks consisting of a single stroke or a rhythmically repeating movement, we focus on finite horizons of length T with episodic restarts [15]. While the policy representation is substantially different, the rhythmic movement resembles a repeated episodic movement in the reinforcement learning process. The general goal in reinforcement learning is to optimize the *expected return* of the policy with parameters $\boldsymbol{\theta}$ defined by

$$J(\boldsymbol{\theta}) = \textstyle\int_{\mathbb{T}} p(\boldsymbol{\tau}) R(\boldsymbol{\tau}) d\boldsymbol{\tau}, \tag{14}$$

where $\boldsymbol{\tau} = [\mathbf{s}_{1:T+1}, \mathbf{a}_{1:T}]$ denotes a sequence of states $\mathbf{s}_{1:T+1} = [\mathbf{s}_1, \mathbf{s}_2, \ldots, \mathbf{s}_{T+1}]$ and actions $\mathbf{a}_{1:T} = [\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_T]$, the probability of an episode $\boldsymbol{\tau}$ is denoted by $p(\boldsymbol{\tau})$ and $R(\boldsymbol{\tau})$ refers to the return of an episode $\boldsymbol{\tau}$ and $\mathbb{T}$ is the set of all possible paths. Using Markov assumption, we can write the path distribution as $p(\boldsymbol{\tau}) = p(\mathbf{s}_1) \prod_{t=1}^{T+1} p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t) \pi(\mathbf{a}_t|\mathbf{s}_t, t)$ where $p(\mathbf{s}_1)$ denotes the initial state distribution and $p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)$ is the next state distribution conditioned on last state and action. Similarly, if we assume additive, accumulated rewards, the return of a path is given by $R(\boldsymbol{\tau}) = \frac{1}{T} \sum_{t=1}^{T} r(\mathbf{s}_t, \mathbf{a}_t, \mathbf{s}_{t+1}, t)$, where $r(\mathbf{s}_t, \mathbf{a}_t, \mathbf{s}_{t+1}, t)$ denotes the immediate reward.

While episodic Reinforcement Learning (RL) problems with finite horizons are common in motor control, few methods

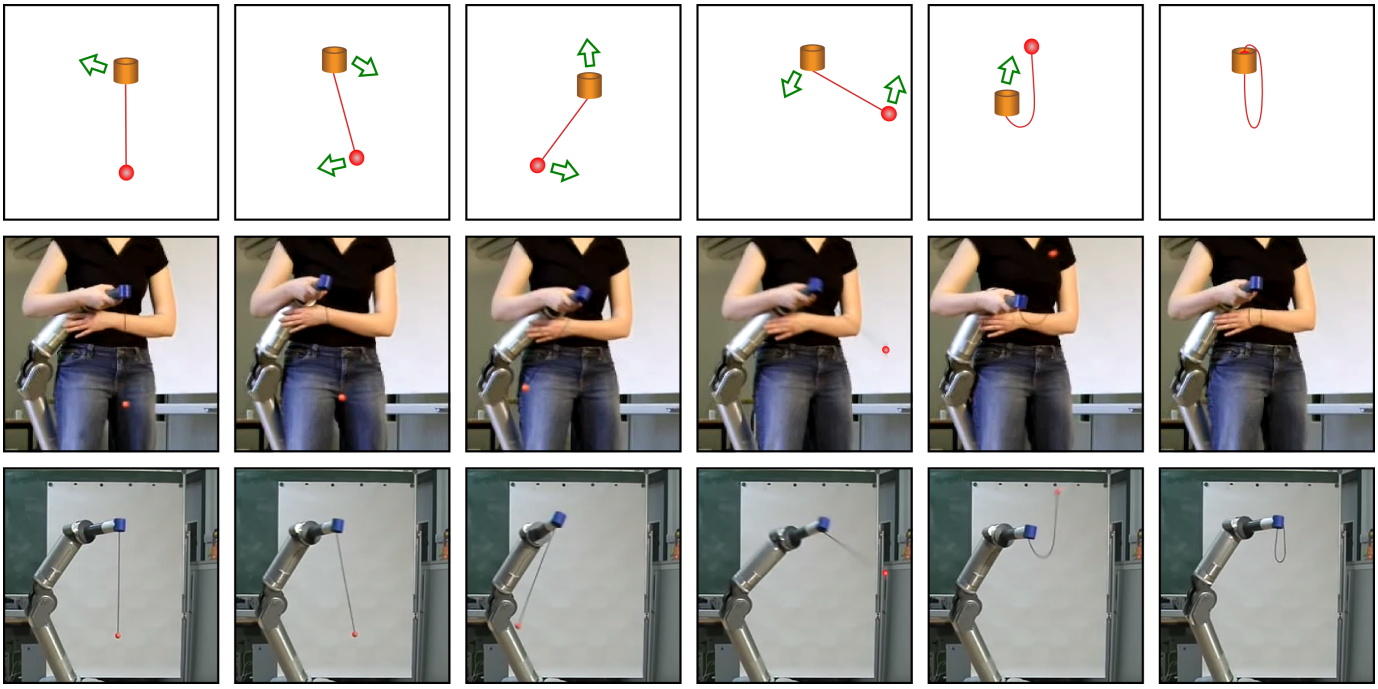

Figure 1. This figure shows schematic drawings of the *Ball-in-a-Cup* motion, a kinesthetic teach-in as well as the performance of the robot after both imitation and reinforcement learning. The green arrows indictate the directions of the current movements in the respective frame for the schematic. The human demonstration was taught to the robot by imitation learning with 31 parameters per joint for an approximately three seconds long trajectory. The robot manages to reproduce the imitated motion quite accurately but the ball misses the cup by approximately 13 centimeters. After around 42 trial runs of our Policy learning by Weighting Exploration with the Returns (PoWER) algorithm, the robot has improved its motion such that the ball goes into the cup for the first time. After roughly 75 rollouts, we have good performance and at the end of the 100 rollouts we have virtually no failures anymore.

exist in the RL literature (notable exceptions are model-free method such as Episodic REINFORCE [17] and the Episodic Natural Actor-Critic eNAC [6] as well as model-based methods, e.g., using differential-dynamic programming [18]). In order to avoid learning of complex models, we focus on model-free methods and, to reduce the number of open parameters, we rather use a novel Reinforcement Learning algorithm which is based on expectation-maximization. Our new algorithm is called Policy learning by Weighting Exploration with the Returns (PoWER) and can be derived from the same higher principle as previous policy gradient approaches, see [12] for details.

2) Policy learning by Weighting Exploration with the Returns (PoWER): When learning motor primitives, we intend to learn a deterministic mean policy $\bar{\mathbf{a}} = \boldsymbol{\theta}^{\mathrm{T}}\boldsymbol{\mu}(\mathbf{s}) = \mathbf{f}(\mathbf{z})$ which is linear in parameters $\boldsymbol{\theta}$ and augmented by additive exploration $\boldsymbol{\varepsilon}(\mathbf{s}, t)$ in order to make model-free reinforcement learning possible. As a result, the explorative policy can be given in the form $\mathbf{a} = \boldsymbol{\theta}^{\mathrm{T}}\boldsymbol{\mu}(\mathbf{s}, t) + \boldsymbol{\epsilon}(\boldsymbol{\mu}(\mathbf{s}, t))$. Previous work in [6], [16], with the notable exception of [19], has focused on state-independent, white Gaussian exploration, i.e., $\boldsymbol{\epsilon}(\boldsymbol{\mu}(\mathbf{s}, t)) \sim \mathcal{N}(0, \Sigma)$, and has resulted into applications such as T-Ball batting [6] and constrained movement [9]. However, from our experience, such unstructured exploration at every step has several disadvantages, i.e., (i) it causes a large variance in parameter updates which grows with the number of time-steps, (ii) it perturbs actions too frequently, as the system acts as a low pass filter the perturbations avarage out and thus, their effects are 'washed' out and (iii) can damage the system executing the trajectory.

Alternatively, as introduced by [19], one could generate a form of structured, state-dependent exploration $\boldsymbol{\epsilon}(\boldsymbol{\mu}(\mathbf{s}, t)) = \boldsymbol{\varepsilon}_t^{\mathrm{T}}\boldsymbol{\mu}(\mathbf{s}, t)$ with $[\boldsymbol{\varepsilon}_t]_{ij} \sim \mathcal{N}(0, \sigma_{ij}^2)$, where σ_{ij}^2 are meta-parameters of the exploration that can be optimized in a similar manner. Each σ_{ij}^2 corresponds to one θ_{ij}. This argument results into the policy $\mathbf{a} \sim \pi(\mathbf{a}_t|\mathbf{s}_t, t) = \mathcal{N}(\mathbf{a}|\boldsymbol{\mu}(\mathbf{s}, \mathbf{t}), \hat{\boldsymbol{\Sigma}}(\mathbf{s}, t))$. This form of policies improves upon the shortcomings of directly perturbed policies mentioned above. Based on the EM updates for Reinforcement Learning as suggested in [12], [16], we can derive the update rule

$$\boldsymbol{\theta}' = \boldsymbol{\theta} + \frac{E_{\boldsymbol{\tau}}\left\{\sum_{t=1}^{T} \boldsymbol{\varepsilon}_t Q^{\pi}(\mathbf{s}_t, \mathbf{a}_t, t)\right\}}{E_{\boldsymbol{\tau}}\left\{\sum_{t=1}^{T} Q^{\pi}(\mathbf{s}_t, \mathbf{a}_t, t)\right\}}, \tag{15}$$

where

$$Q^{\pi}(\mathbf{s}, \mathbf{a}, t) = E\left\{\sum_{\tilde{t}=t}^{T} r\left(\mathbf{s}_{\tilde{t}}, \mathbf{a}_{\tilde{t}}, \mathbf{s}_{\tilde{t}+1}, \tilde{t}\right) | \mathbf{s}_t = \mathbf{s}, \mathbf{a}_t = \mathbf{a}\right\}$$

is the state-action value function. Note that this algorithm does not need the learning rate as a meta-parameter.

In order to reduce the number of trials in this on-policy scenario, we reuse the trials through importance sampling [15], [20]. To avoid the fragility sometimes resulting from importance sampling in reinforcement learning, samples with very small importance weights are discarded.

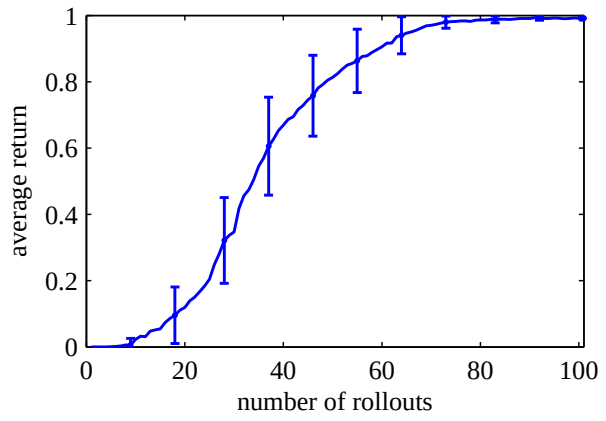

Figure 2. This figure shows the expected return of the learned policy in the Ball-in-a-Cup evaluation averaged over 20 runs.

The more shape parameters $\mathbf{w}$ are used the more details can be captured in a motor primitive and it can ease the imitation learning process. However, if the motor primitives need to be refined by RL, each additional parameter slows down the learning process. The parameters σ_{ij}^2 determine the exploration behavior where larger values lead to greater changes in the mean policy and, thus, may lead to faster convergence but can also drive the robot in unsafe regimes. The optimization of the parameters decreases the exploration during convergence.

IV. Robot Evaluation

The methods presented in this paper are evaluated on two learning problems on a real, seven degree of freedom Barrett WAM, i.e., we learn the discrete task of *Ball-in-a-Cup* and the rhythmic task *Ball-Paddling*. The resulting simplicity and speed of the learning process demonstrate the suitability of the motor primitive-based learning framework for practical application.

The kinematic trajectory output by the motor primitives is executed using a computed torque controller based on a ridgid body dynamics model with parameters estimated from recorded data and an additional friction model. Gains are set low so that the robot is relatively compliant. This may lead to execution errors in the kinematic trajectory. However, Reinforcement Learning will directly learn policies that take into account these imperfections of the controller.

A. Discrete Movement: Ball-in-a-Cup

The children motor game Ball-in-a-Cup, also known as Balero and Bilboquet [21] is challenging even for a grown up. The toy has a small cup which is held in one hand (or, in our case, is attached to the end-effector of the robot) and the cup has a small ball hanging down on a string (the string has a length of 40cm for our toy). Initially, the ball is hanging down vertically in a rest position. The player needs to move fast in order to induce a motion in the ball through the string, toss it up and catch it with the cup, a possible movement is illustrated in Figure 1 in the top row.

Note that learning Ball-in-a-Cup and Kendama have previously been studied in robotics and we are going to contrast a few of the approaches here. While we learn directly in the joint space of the robot, Takenaka et al. [22] recorded planar human cup movements and determined the required joint movements for a planar, three degree of freedom (DoF) robot so that it could follow the trajectories while visual feedback was used for error compensation. Both Sato et al. [23] and Shone [24] used motion planning approaches which relied on very accurate models of the ball while employing only one DoF in [24] or two DoF in [23] so that the complete state-space could be searched exhaustively. Interestingly, exploratory robot moves were used in [23] to estimate the parameters of the employed model. The probably most advanced preceding work on learning Kendama was done by Miyamoto [25] who used a seven DoF anthropomorphic arm and recorded human motions to train a neural network to reconstruct via-points. Employing full kinematic knowledge, the authors optimize a desired trajectory.

The state of the system can be described by joint angles and joint velocities of the robot as well as the the Cartesian coordinates and velocities of the ball. The actions are the joint space accelerations where each of the seven joints is driven by a separate motor primitive with one common canonical system. The movement uses all seven degrees of freedom and is not on a plane. All motor primitives are perturbed separately but employ the same joint final reward. At the time t_c where the ball passes the rim of the cup with a downward direction, we compute the reward as $r(t_c) = \exp(-\alpha(x_c - x_b)^2 - \alpha(y_c - y_b)^2)$ while we have $r(t) = 0$ for all $t = t_c$. Here, the cup position is denoted by $[x_c, y_c, z_c] \in \mathbb{R}^3$, the ball position $[x_b, y_b, z_b] \in \mathbb{R}^3$ and we have a scaling parameter $\alpha = 100$. The directional information is necessary as the algorithm could otherwise learn to hit the bottom of the cup with the ball. The reward is not only affected by the movements of the cup but foremost by the movements of the ball which are sensitive to small changes in the movement. A small perturbation of the initial condition or during the trajectory can change the movement of the ball and hence the outcome of the complete movement. The position of the ball is estimated using a stereo vision system and needed to determine the reward.

Due to the complexity of the task, Ball-in-a-Cup is even a hard motor task for children who usually only succeed after observing another person presenting a demonstration first, and after subsequent trial-and-error-based learning. Mimicking how children learn to play Ball-in-a-Cup, we first initialize the motor primitives by imitation and, subsequently, improve them by reinforcement learning.

We recorded the motions of a human player by kinesthetic teach-in in order to obtain an example for imitation as shown in Figure 1 (middle row). Kinesthetic teach-in means "taking the robot by the hand", performing the task by moving the robot while it is in gravity-compensation mode and recording the joint angles, velocities and accelerations. A single demonstration was used for imitation learning. Learning from mulitple demonstrations did not improve the performance as

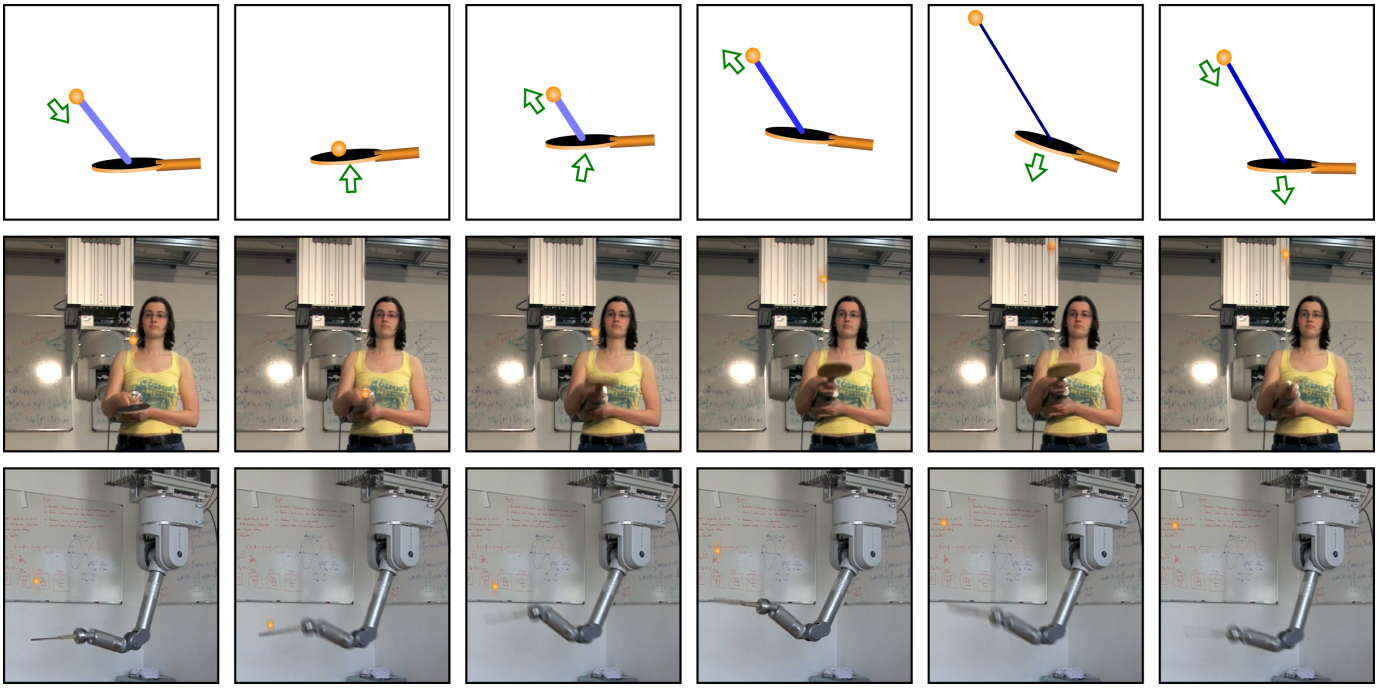

Figure 3. This figure shows schematic drawings of the *Ball-Paddling* motion, a kinesthetic teach-in as well as the performance of the robot after imitation learning. When the string is stretched it is shown as thinner and darker. The human demonstration was taught to the robot by imitation learning with 10 parameters per joint for the rhythmic motor primitive. An additional discrete motor primitive is used for the start-up phase. Please see Section 4.2 and the accompanying video for details.

the task is sensitive to small details. As expected, the robot fails to reproduce the presented behavior even if we use all the recorded details for the imitation. Thus, reinforcement learning is needed for self-improvement. The more parameters are used for the learning, the slower is the convergence. We found that a necessary condition for convergence with the reward we defined is that the ball goes above the rim of the cup so that the algorithm gets at least a small positive reward for all rollouts. In this way the algorithm can compare the performance of the different rollouts. We determined that 31 shape-parameters per motor primitive are needed.

In [12] we benchmarked our novel algorithm and several widely used algorithms on tasks having characteristics similar to this one. As a result we employ our best algorithm, PoWER. The meta-parameters σ_{ij} are initially set in the order of magnitude of the median of the parameters for each motor primitive and are then optimized alongside the shape-parameters by PoWER. The performance of the algorithm is fairly robust for values chosen in this range.

Figure 2 shows the expected return over the number of rollouts where convergence to a maximum is clearly recognizable. The robot regularly succeeds at bringing the ball into the cup after approximately 75 rollouts. A nine year old child got the ball in the cup for the first time after 35 trials while the robot got the ball in for the first time after 42 rollouts. However, after 100 trials, the robot exhibits perfect runs in every single trial while, from our experience, the child does not have a comparable success rate. Of course, such a comparison with a child is contrived as a robot can precisely reproduce movements unlike any human being and that children can most likely adapt faster to changes in the setup.

After 100 rollouts, the meta-parameters such as the exploration rate have converged to negligible size and do not influence the outcome of the behavior any longer. The experiments in this paper use the original variant of the motor primitives which cannot deal with large perturbations, however, the extended variable-feedback variant presented in [5] can deal with a variety of changes, e.g., in the length of the string, the size or weight of the ball, directly while the approach presented in this paper will recover quickly by learning an adjusted policy in a few roll-outs. In [5] we also show that learning a strategy of pulling the ball up and moving the cup under the ball (as in Kendama) is possible in approximately the same number of trials. In simulation, we have discussed a variety of different strategies for Ball-in-a-Cup in [26].

B. Rhythmic Movement with start-up phase: Ball-Paddling

In Ball-Paddling, we have a table-tennis ball that is attached to a table-tennis paddle by an elastic string. The goal is to have the ball bouncing above the paddle. The string avoids that the ball is falling down but also pulls the ball back towards the center of the paddle if the ball is hit sufficiently hard (i.e., the string is also stretched sufficiently as a consequence). The task is fairly easy to perform open-loop once the player has determined appropriate amplitude and frequency for the motion. Furthermore, the task is robust to small changes of these parameters as well as to small perturbations of the environment. We again recorded the motions of a human

player using kinesthetic teach-in in order to obtain a demonstration for imitation learning as shown in Figure **3**. From the imitation, it can be determined by cross-validation that 10 shape-parameters per motor primitive are sufficient. The shape parameters, the amplitude and the period of the motion are estimated from the demonstration after the start-up phase.

However, as we start with a still robot where the ball rests on the paddle, we require a start-up phase in order to perform the task successfully. This initial motion has to induce more energy in order to get the motion started and to extend the string sufficiently. Once the ball falls below the paddle, the rhythmic motion becomes chaotic and the behavior cannot be recovered without an additional still phase – the same was true for all human presenters. For our setup, the start-up phase consists (as exhibited by the teacher's movements) of moving the paddle slower and further up than during the rhythmic behavior. This kind of movement can easily be achieved in the dynamic systems motor primitives framework by imposing another discrete dynamical systems primitive that gradually adapts the period parameter τ globally and the amplitude modifier a_k to the ones encountered in the rhythmic behavior. The discrete modifier motor primitive is applied additively to the two parameters. The goal parameter of this modifier primitive is zero and thus, its influence vanishes after the initialization time τ_{modif}. With this start-up phase, imitation learning from demonstrations suffices to reproduce the motor skill successfully. To our knowledge, this application is probably the first where both rhythmic and discrete dynamic systems primitives are used together to achieve a particular task.

V. Conclusion

In this paper, we present both novel learning algorithms and experiments using the dynamic systems motor primitive [2], [4]. For doing so, we have first described this representation of the motor primitives. Subsequently, we have discussed both appropriate imitation learning methods by locally weighted regression and derived our currently best-suited reinforcement learning algorithm for this framework, i.e., Policy learning by Weighting Exploration with the Returns (PoWER). We show that two complex motor tasks, i.e., *Ball-in-a-Cup* and *Ball-Paddling*, can be learned on a real, physical Barrett WAM using the methods presented in this paper. Of particular interest is the Ball-Paddling application as it requires the combination of both rhythmic and discrete dynamic systems primitives in order to achieve a particular task.

References

[1] T. Flash and B. Hochner, "Motor primitives in vertebrates and invertebrates," *Current Opinions in Neurobiology*, vol. 15, pp. 660–666, 2005.

[2] A. J. Ijspeert, J. Nakanishi, and S. Schaal, "Learning attractor landscapes for learning motor primitives," in *Advances in Neural Information Processing Systems (NIPS)*, S. Becker, S. Thrun, and K. Obermayer, Eds., vol. 15. Cambridge, MA: MIT Press, 2003, pp. 1547–1554.

[3] S. Schaal, J. Peters, J. Nakanishi, and A. J. Ijspeert, "Control, planning, learning, and imitation with dynamic movement primitives," in *Proceedings of the Workshop on Bilateral Paradigms on Humans and Humanoids, IEEE 2003 International Conference on Intelligent RObots and Systems (IROS)*, 2003.

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

[5] J. Kober, B. Mohler, and J. Peters, "Learning perceptual coupling for motor primitives," in *Proceedings of the IEEE/RSJ 2008 International Conference on Intelligent RObots and Systems (IROS)*, 2008, pp. 834–839.

[6] J. Peters and S. Schaal, "Policy gradient methods for robotics," in *Proceedings of the IEEE/RSJ 2006 International Conference on Intelligent RObots and Systems (IROS)*, 2006, pp. 2219 – 2225.

[7] D. Pongas, A. Billard, and S. Schaal, "Rapid synchronization and accurate phase-locking of rhythmic motor primitives," in *Proceedings of the IEEE 2005 International Conference on Intelligent RObots and Systems (IROS)*, 2005, pp. 2911–2916.

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

[9] 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, pp. 1521–1544, 2007.

[10] H. Urbanek, A. Albu-Schäffer, and P.v.d.Smagt, "Learning from demonstration repetitive movements for autonomous service robotics," in *Proceedings of the IEEE/RSL 2004 International Conference on Intelligent RObots and Systems (IROS)*, 2004, pp. 3495–3500.

[11] P. Abbeel and A. Y. Ng, "Apprenticeship learning via inverse reinforcement learning," in *Proceedings of the Twenty-First International Conference on Machine Learning (ICML)*. ACM, 2004.

[12] J. Kober and J. Peters, "Policy search for motor primitives in robotics," in *Advances in Neural Information Processing Systems (NIPS)*, 2008.

[13] D.-H. Park, H. Hoffmann, P. Pastor, and S. Schaal, "Movement reproduction and obstacle avoidance with dynamic movement primitives and potential fields," in *IEEE International Conference on Humanoid Robots (HUMANOIDS)*, 2008.

[14] R. E. Bellman, *Dynamic Programming*. Princeton, NJ: Princeton University Press, 1957.

[15] R. Sutton and A. Barto, *Reinforcement Learning*. MIT PRESS, 1998.

[16] J. Peters and S. Schaal, "Reinforcement learning for operational space," in *Proceedings of the International Conference on Robotics and Automation (ICRA)*, Rome, Italy, 2007.

[17] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Machine Learning*, vol. 8, pp. 229–256, 1992.

[18] C. G. Atkeson, "Using local trajectory optimizers to speed up global optimization in dynamic programming," in *Advances in Neural Information Processing Systems 6 (NIPS)*, J. E. Hanson, S. J. Moody, and R. P. Lippmann, Eds. Morgan Kaufmann, 1994, pp. 503–521.

[19] T. Rückstieß, M. Felder, and J. Schmidhuber, "State-dependent exploration for policy gradient methods," in *Proceedings of the European Conference on Machine Learning (ECML)*, 2008, pp. 234–249.

[20] C. Andrieu, N. de Freitas, A. Doucet, and M. I. Jordan, "An introduction to MCMC for machine learning," *Machine Learning*, vol. 50, no. 1, pp. 5–43, 2003.

[21] Wikipedia, "Ball-in-a-cup," January 2009. [Online]. Available: http://en.wikipedia.org/wiki/Ball_in_a_cup

[22] K. Takenaka, "Dynamical control of manipulator with vision : "cup and ball" game demonstrated by robot," *Transactions of the Japan Society of Mechanical Engineers. C*, vol. 50, no. 458, pp. 2046–2053, 1984.

[23] S. Sato, T. Sakaguchi, Y. Masutani, and F. Miyazaki, "Mastering of a task with interaction between a robot and its environment : "kendama" task," *Transactions of the Japan Society of Mechanical Engineers. C*, vol. 59, no. 558, pp. 487–493, 1993.

[24] T. Shone, G. Krudysz, and K. Brown, "Dynamic manipulation of kendama," Research Project, Rensselaer Polytechnic Institute, Tech. Rep., 2000.

[25] H. Miyamoto, S. Schaal, F. Gandolfo, H. Gomi, Y. Koike, R. Osu, E. Nakano, Y. Wada, and M. Kawato, "A kendama learning robot based on bi-directional theory," *Neural Networks*, vol. 9, no. 8, pp. 1281–1302, 1996.

[26] S. Chiappa, J. Kober, and J. Peters, "Using bayesian dynamical systems for motion template libraries," in *Advances in Neural Information Processing Systems (NIPS)*. Cambridge, MA: MIT Press, 2009.