Imitation and Reinforcement Learning

Practical Learning Algorithms for Motor Primitives in Robotics

by Jens Kober and Jan Peters

To date, most robots are still programmed by a smart operator who uses human understanding of the desired task 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 had considered. For example, human programming has become the main bottleneck for manufacturing of low-cost products in low numbers. This problem could be alleviated by robots that can learn new skills and improve their existing abilities autonomously. However, off-the-shelf machine learning techniques do not scale to high-dimensional, anthropomorphic robots. Instead, robot learning requires methods that employ both representations and algorithms appropriate for this domain. When humans learn new motor skills, e.g., paddling a ball with a table-tennis racket, throwing darts, or hitting a tennis ball, it is highly likely that they rely on a small set of motor primitives (MPs) and use imitation as well as reinforcement learning (RL) [1]. Inspired by this example, we will discuss the technical counterparts in this article 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 MPs was put forward by Ijspeert et al. [2] as a general approach of representing control policies for basic movements. The resulting movement generation has a variety of favorable properties, i.e., basic stability properties, the ability to encode either single-stroke or rhythmic behaviors, as well as rescalability with respect to time, goal, and amplitude. As this representation is linear in its parameters, learning can be sufficiently fast for real-time applications in robotics. A series of such frameworks has been introduced to date [2], [3]. Previous applications include a variety of different basic motor skills such as tennis swings [2], T-ball batting [4], drumming [5], planar biped walking [6], constrained reaching tasks [7] and even in tasks with potential industrial application [8]. Nevertheless, most of the previous work in MP learning (with the exceptions of [4] and [7]) has focused on learning by imitation without subsequent self-improvement. Please refer to [9] for a review of imitation learning methods. In real life, a human demonstration is usually not perfect nor does it suffice for near-optimal performance. Thus, additional RL is essential for both performance-based refinement and continuous adaptation of the presented skill. Note that this approach is inherently different from the complementary idea of apprenticeship learning [10], which attempts to infer the intent of the teacher and learn his policy through an inverse RL approach.

In this article, we discuss several practical recipes for imitation learning and RL of MPs used in our current best performing setups. We attempt to present the methods with sufficient detail such that the results can be transferred to industrial applications by the robotics practitioners. As a first step, we review the MP framework in its current formulation [11] and discuss learning of discrete, single-stroke movements as well as rhythmic, repeated movements. We show how combination of imitation learning and RL can be used for learning tasks such as ball-in-a-cup on a real Barrett whole arm manipulator (WAM). In contrast to previous work on rhythmic MPs [2], [11], we have created a novel formulation of the rhythmic MPs that includes a transient start-up phase required in many practical applications. An example of such a task is ball paddling where a ball on a string needs to be brought into a stable limit cycle by a rhythmic movement with a start-up phase. The accompanying video shows the human demonstration and the learned performance for both tasks (see http://robot-learning.de/Research/LearningMotorPrimitives).

Learning Methods for MPs

The generic idea of encoding elementary movements using dynamical systems has become increasingly accepted in the motor skill learning community. A major breakthrough was the suggestion in [2] to use two kinds of dynamical systems with a one-way, parametrized connection such that one system drives the others (see Figure 1). As a result of this prestructuring, the system is guaranteed to be stable, and the unstable or chaotic

• *Address for Correspondence: Jens Kober, Department of Empirical Inference and Machine Learning, Max Planck Institute for Biological Cybernetics, 72076 Tübingen, Germany.*
E-mail: kober@tuebingen.mpg.de

Manuscript received Oct 15, 2009.; revised Feb 1, 2010

behaviors common in earlier learning systems, which attempted to learn the complete dynamical systems, can be avoided. Please note that using this stable dynamical system as reference does not guarantee that the full robot system remains stable. If the parametrized connection is a function that is linear in the parameters, it becomes straightforward to learn. This partition leads to two components. There is a canonical, hidden system

$$\dot{z} = u\left(z\right) \tag{1}$$

that acts as an adjustable clock or phase of the movement with state z. As illustrated in Figure 1, this canonical system u drives the second component, the transformed systems

$$\dot{\mathbf{x}} = \mathbf{v}\left(\mathbf{x}, z, \mathbf{g}, \boldsymbol{\theta}\right), \tag{2}$$

for all considered degrees of freedom (DoFs) i, where $\boldsymbol{\theta}$ denotes the internal parameter needed for learning. The state of the transformed systems $\mathbf{x} = [\mathbf{x}_1, \mathbf{x}_2]$ consists of desired positions $\mathbf{q} = \mathbf{x}_1$ and velocities $\dot{\mathbf{q}} = \tau\mathbf{x}_2$ of the robot system either in joint space or in task space. The accelerations can also be determined by $\ddot{\mathbf{q}} = \tau\dot{\mathbf{x}}_2$ using the dynamical system. A suitable controller is used to convert these into motor torques.

This approach has a multitude of advantages [2] as it can ensure the stability of the movement as the one-way connection is inherently stable if the separated dynamical systems are stable. While it may represent arbitrary movements, it can be generated with well-understood components, such as linear systems used in conjunction with established function approximation. Employing the dynamical systems in (1) and (2) allows to determine the stability of the movement, choosing between a rhythmic and a discrete movement and rescaling in both time and movement amplitude. Feedback terms can also be added as in [5] and [12]. With the right choice of function approximator, fast learning from a teacher's presentation is possible.

Case Study: Discrete MP Policies

To make this approach more concrete, we review the MP policy representation with the example of a single discrete movement. Since its inception in [2], the representation has been simplified and it can be shown that a single, first order system $\dot{z} = -\tau\alpha_z z$ suffices as the canonical system [11]. This system can be used to directly adjust the duration of the movement with time constant $\tau = 1/T$, where T is the duration of an MP. Parameter α_z is chosen such that $z \approx 0$ at T to ensure safe movement termination. We can express the MP function $\mathbf{v}$ in the following form:

$$\dot{\mathbf{x}}_2 = \tau\alpha_x\left(\beta_x\left(\mathbf{g} - \mathbf{x}_1\right) - \mathbf{x}_2\right) + \tau\mathbf{A}\mathbf{f}\left(z\right), \tag{3}$$

$$\dot{\mathbf{x}}_1 = \tau\mathbf{x}_2. \tag{4}$$

This set of differential equations has the same time constant τ as the canonical system. The parameters α_x, β_x are set such that the system is critically damped when $\mathbf{A} = \mathbf{0}$. The remaining parameters are the goal $\mathbf{g}$, a transformation function $\mathbf{f}$, and a diagonal matrix $\mathbf{A}$ that acts as an amplitude modifier. The value $\mathbf{g} - \mathbf{x}_1^0$, with initial position $\mathbf{x}_1^0$, is a common choice [11] for the diagonal elements of $\mathbf{A}$, as it ensures linear rescaling under goal variations. However, other choices may be better suited for specific tasks (e.g., for reaching tasks, the solution in [12] is often preferable). The transformation function

$$\mathbf{f}\left(z\right) = \textstyle\sum_{n=1}^{N}\psi^n\left(z\right)\boldsymbol{\theta}^n z. \tag{5}$$

alters the output of the transformed system in (3) so that it can represent complex nonlinear patterns. Here $\boldsymbol{\theta}^n$ contains the nth set of adjustable parameters for all

```
MP Policy:
Set hyperparameters: constants $\alpha_x$ and $\beta_x$ such that the transformed system is critically damped, number of shape parameters $N$, widths $\mathbf{h}$, and centers $\mathbf{c}$ for the shape parameters
1) for discrete MPs: decay of the canonical system $\alpha_z$
2) for rhythmic MPs: phase rate of change $\omega$
Set parameters: duration modifier $\tau$, shape parameters $\boldsymbol{\theta}_n$ for $n = \{1, \ldots, N\}$, and amplitude modifier $\mathbf{A}$
1) for discrete MPs: final position $\mathbf{g}$
2) for rhythmic MPs: mean position $\mathbf{m}$
Set initial values: initial position $\mathbf{x}_1 = \mathbf{q}_0$ and initial velocity $\mathbf{x}_2 = \dot{\mathbf{q}}_0/\tau$
1) for discrete MPs: phase variable $z = 1$
2) for rhythmic MPs: phase variable $z = 0$
For each time step
    Evaluate canonical system
    1) for discrete MPs: $\dot{z} = -\tau\alpha_z z$
    2) for rhythmic MPs: $\dot{z} = \tau\omega$
    Evaluate transformation function
    1) for discrete MPs: $\mathbf{f}(z) = \sum_{n=1}^{N}\psi^n(z)\boldsymbol{\theta}^n z$ with
       $\psi^n = \frac{\exp\left(-h_n^{-2}(z-c_n)^2\right)}{\sum_{m=1}^{N}\exp\left(-h_m^{-2}(z-c_m)^2\right)}$
    2) for rhythmic MPs: $\mathbf{f}(z) = \sum_{n=1}^{N}\psi^n(z)\boldsymbol{\theta}^n$ with
       $\psi^n = \frac{\exp\left(-h_n^{-2}(1-\cos(z-c_n))\right)}{\sum_{m=1}^{N}\exp\left(-h_m^{-2}(1-\cos(z-c_m))\right)}$
    Evaluate transformed system:
    1) for discrete MPs:
       $\dot{\mathbf{x}}_2 = \tau\alpha_x(\beta_x(\mathbf{g}-\mathbf{x}_1)-\mathbf{x}_2)+\tau\mathbf{A}\mathbf{f}(z)$ and $\dot{\mathbf{x}}_1 = \tau\mathbf{x}_2$
    2) for rhythmic MPs:
       $\dot{\mathbf{x}}_2 = \tau\alpha_x(\beta_x(\mathbf{m}-\mathbf{x}_1)-\mathbf{x}_2)+\tau\mathbf{A}\mathbf{f}(z)$ and $\dot{\mathbf{x}}_1 = \tau\mathbf{x}_2$
    Numerically integrate the canonical and transformed system and obtain the desired joint positions $\mathbf{q} = \mathbf{x}_1$, velocities $\dot{\mathbf{q}} = \dot{\mathbf{x}}_1$, and accelerations $\ddot{\mathbf{q}} = \tau\dot{\mathbf{x}}_2$.
```

TABLE 1
Representing movements with dynamical systems

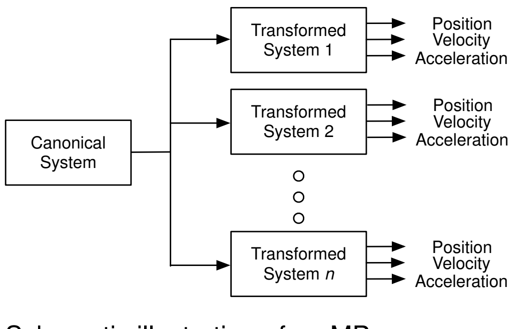

Fig. 1. Schematic illustration of an MP.

```
Recipe for Imitating Movements
For each DoF $i$ and each parameter $n$ separately
    Extract from reference trajectory $\mathbf{q}$, $\dot{\mathbf{q}}$, $\ddot{\mathbf{q}}$
    1) for discrete MPs: duration $T$ and final position $g = q_T$
    2) for rhythmic MPs: period $T$ and mean position $m = \mathrm{mean}(\mathbf{q})$

    Set amplitude modifier $[\mathbf{A}]_{ii} = g - q_0$, time-scale $\tau = 1/T$
    as well as the hyperparameters: constants $\alpha_x$ and $\beta_x$,
    number of shape parameters $N$, widths $\mathbf{h}$ and centers $\mathbf{c}$
    1) for discrete MPs: decay of the canonical system $\alpha_z$
    2) for rhythmic MPs: phase rate of change $\omega$

    Calculate the values $z_t$ of the basis function and the values
    $\psi_t^n$ of the weight for all samples $j$ using
    1) for discrete MPs: $\dot{z} = -\tau\alpha_z z$ and
       $\psi^n = \frac{\exp\left(-h_n^{-2}(z-c_n)^2\right)}{\sum_{m=1}^N \exp\left(-h_m^{-2}(z-c_m)^2\right)}$
    2) for rhythmic MPs: $\dot{z} = \tau\omega$ and
       $\psi^n = \frac{\exp\left(-h_n^{-2}(1-\cos(z-c_n))\right)}{\sum_{m=1}^N \exp\left(-h_m^{-2}(1-\cos(z-c_m))\right)}$

    Calculate from the parameters and the reference trajectory
    $q_t$, $\dot{q}_t$, $\ddot{q}_t$ the reference transformation function $f_t^{\mathrm{ref}}$
    1) for discrete MPs:
       $f_t^{\mathrm{ref}} = \ddot{q}_t/(\tau^2[\mathbf{A}]_{ii}) - \alpha_x\left(\beta_x\left(g - q_t\right) - \dot{q}_t/\tau\right)/[\mathbf{A}]_{ii}$
    2) for rhythmic MPs:
       $f_t^{\mathrm{ref}} = \ddot{q}_t/(\tau^2[\mathbf{A}]_{ii}) - \alpha_x\left(\beta_x\left(m - q_t\right) - \dot{q}_t/\tau\right)/[\mathbf{A}]_{ii}$

    Create matrices from the basis function $[\mathbf{Z}]_t = z_t$, weights
    $\mathbf{\Psi} = \mathrm{diag}\left(\psi_1^n, \ldots, \psi_t^n, \ldots, \psi_T^n\right)$ and reference values
    $[\mathbf{f}^{\mathrm{ref}}]_t = f_t^{\mathrm{ref}}$

    Perform regression $\theta^n = \left(\mathbf{Z}^{\mathrm{T}}\mathbf{\Psi}\mathbf{Z}\right)^{-1}\mathbf{Z}^{\mathrm{T}}\mathbf{\Psi}\mathbf{f}^{\mathrm{ref}}$ to learn the
    $n$th shape parameter
Return parameters
```

TABLE 2
Imitation of discrete and rhythmic movements

DoFs, N is the number of parameters per DoF, and $\psi^n(z)$ are the corresponding weights. Normalized Gaussian kernels are used as weights, given by

$$\psi^n = \frac{\exp\left(-h_n^{-2}\left(z-c_n\right)^2\right)}{\sum_{m=1}^N \exp\left(-h_m^{-2}\left(z-c_m\right)^2\right)}. \tag{6}$$

Such functions localize the interaction in phase space using the centers c_n and widths h_n. All DoFs are synchronous as the dynamical systems for all DoFs start at the same time, have the same duration, and the shape of the movement is generated using the z-dependant transformation $\mathbf{f}\left(z\right)$ in (5).

The recipe for complete instantiation of MP policies for either rhythmic or discrete movements is given in Table 1.

Imitation Learning by Weighted Regression

Many movements can be learned [2], [11], or at least initialized [4], [7] using imitation learning. This step can be performed efficiently in the context of dynamical systems MPs, as the transformation function (5) is linear in its parameters [2], [11]. The policy parameters are estimated separately for each DoF i. We omit the index i in this section for better readability. The weighted squared error $e_n^2 = \sum_{t=1}^T \psi_t^n (f_t^{\mathrm{ref}} - z_t\theta^n)^2$ is the appropriate cost function in this context, which needs to be minimized for all parameters θ^n. To avoid undercompleteness and avoid overfitting, the number of parameters n should be significantly smaller than the number of training points T. Here, the corresponding weighting functions are denoted by ψ_t^n and the basis function by z_t. A good starting point for the weighting functions are centers c_n equispaced in time and a width h_n of two third the distance between the centers. The number of weighting functions as well as the centers and widths can be optimized by cross-validation. The reference or target signal f_t^{ref} is the desired transformation function, and j indicates the sample. The error can thus be rewritten in matrix form as

$$e_n^2 = \left(\mathbf{f}^{\mathrm{ref}} - \mathbf{Z}\theta^n\right)^{\mathrm{T}}\mathbf{\Psi}\left(\mathbf{f}^{\mathrm{ref}} - \mathbf{Z}\theta^n\right) \tag{7}$$

with $\mathbf{f}^{\mathrm{ref}}$ containing the value of f_t^{ref} for all samples j, $\mathbf{\Psi} = \mathrm{diag}\left(\psi_1^n, \ldots, \psi_t^n, \ldots, \psi_T^n\right)$ and $[\mathbf{Z}]_t = z_t$. The resulting weighted linear regression problem can be solved straightforwardly by the unbiased parameter estimator

$$\theta^n = \left(\mathbf{Z}^{\mathrm{T}}\mathbf{\Psi}\mathbf{Z}\right)^{-1}\mathbf{Z}^{\mathrm{T}}\mathbf{\Psi}\mathbf{f}^{\mathrm{ref}}. \tag{8}$$

This general approach was originally 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 MPs was extracted using first repetitions, and again, the time constant τ is set accordingly. The practical method for using such imitation learning approach for either a discrete movement or rhythmic one is given in Table 2.

Most previous work on rhythmic MPs only treated the case when the rhythmic movement is already in the limit cycle [2], [11]. Despite being essential in many tasks, such as in bouncing a ball, the transient has been neglected. Hence, we propose an imitation framework that makes use of the discrete primitive to modulate rhythmic ones

```
Recipe for Imitating Rhythmic Movements with Start-Up
Extract duration $T_{\mathrm{start}}$ of the start-up phase, i.e., the time the
movement needs to converge to a regular rhythmic behavior
For each DoF
    Apply *recipe for imitating movements* for the rhythmic
    movement after the start-up phase, i.e., use $q_t$, $\dot{q}_t$, $\ddot{q}_t$ with
    $j = \{T_{\mathrm{start}}, T_{\mathrm{start}+1}, \ldots, T_{\mathrm{final}}\}$ for the regression yielding
    time-scale $\tau_r$

    Define a system in the form of the discrete canonical
    system that gradually changes the time scale $\tau$ from a start
    value $\tau_0$ to the rhythmic value $\tau_r$ ensuring that the first
    period is stretched to the start-up time $T_{\mathrm{start}}$ and that $\tau$
    corresponds to the rhythmic period $\tau_r$ thereafter

    Run the motor primitive using the changing time-scale $\tau$
    and calculate the difference to the desired behavior $\Delta\mathbf{f}^{\mathrm{ref}}$

    Apply *recipe for imitating movements* to learn a discrete
    motor primitive that compensates the differences $\Delta\mathbf{f}^{\mathrm{ref}}$ if
    added to $\mathbf{f}$ of the rhythmic primitive
Return parameters
```

TABLE 3
Imitation of rhythmic movements with start-up

```
Recipe for RL of MPs
Input: initial policy parameters $\boldsymbol{\theta}_0$
Repeat
    Sample: Perform rollout(s) using action
    $\mathbf{a} = (\boldsymbol{\theta} + \boldsymbol{\varepsilon}_t)^{\mathrm{T}} \boldsymbol{\phi}(\mathbf{s}, t)$ with exploration $[\boldsymbol{\varepsilon}_t]_{ij} \sim \mathcal{N}(0, \sigma_{ij}^2)$ as
    stochastic policy and collect all $(t, \mathbf{s}_t, \mathbf{a}_t, \mathbf{s}_{t+1}, \boldsymbol{\varepsilon}_t, r_{t+1})$ for
    $t = \{1, 2, \dots, T+1\}$.
    Estimate: Use unbiased estimate of the value function
    $\hat{Q}^{\pi}(\mathbf{s}, \mathbf{a}, t) = \sum_{\tilde{t}=t}^{T} r(\mathbf{s}_{\tilde{t}}, \mathbf{a}_{\tilde{t}}, \mathbf{s}_{\tilde{t}+1}, \tilde{t})$.
    Reweight: rollouts, discard low-reward rollouts.
    Update policy using
    $\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k + \left\langle \sum_{t=1}^{T} \boldsymbol{\varepsilon}_t Q^{\pi}(\mathbf{s}, \mathbf{a}, t) \right\rangle \Big/ \left\langle \sum_{t=1}^{T} Q^{\pi}(\mathbf{s}, \mathbf{a}, t) \right\rangle$.
Until convergence $\boldsymbol{\theta}_{k+1} \approx \boldsymbol{\theta}_k$
```

TABLE 4
Self-improvement for MPs

in order to generate a start-up phase for the movement. The resulting algorithm employs only components from this section and is shown in Table 3.

RL of MPs

Optimizing the performance of a system by trial and error is often referred to as RL, see [13], [14], and it is an essential ability for skill-learning systems. However, RL for MPs is a very specific type of learning problem, where it is hard to apply generic algorithms [4], [15] due to the high-dimensional states $\mathbf{s} = [\mathbf{z}, \mathbf{x}]$ that defy discretization of the state space. Hence, we need novel domain-appropriate RL algorithms for parametrized policies in episodic control problems [16]. For self-improvement, it is necessary to add an exploration term $\boldsymbol{\epsilon}$ that modifies the actions. In our case, this results in an output action $\mathbf{a} = \mathbf{f}(\mathbf{z}) + \boldsymbol{\epsilon}$ that depends on the output of our transformation function. As a result, we have a stochastic policy $\mathbf{a} \sim \pi(\mathbf{s})$ with parameters $\boldsymbol{\theta} \in \mathbb{R}^N$ which can be seen as a distribution over the actions given the states. After a time-step Δt, the agent 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, fixed horizons of length T with episodic restarts [13]. While the policy is substantially different, rhythmic movements can still be learned by episodic RL. The goal in RL is usually to optimize the expected return of the policy with parameters $\boldsymbol{\theta}$ defined by

$$J(\boldsymbol{\theta}) = \int_{\boldsymbol{\Xi}} p(\boldsymbol{\xi}) R(\boldsymbol{\xi}) d\boldsymbol{\xi}, \tag{9}$$

where the episode or rollout $\boldsymbol{\xi} = [\mathbf{s}_{1:T+1}, \mathbf{a}_{1:T}]$ denotes a sequence of states $\mathbf{s}_{1:T+1} = [\mathbf{s}_1, \mathbf{s}_2, \dots, \mathbf{s}_{T+1}]$ and actions $\mathbf{a}_{1:T} = [\mathbf{a}_1, \mathbf{a}_2, \dots, \mathbf{a}_T]$, the probability of an episode $\boldsymbol{\xi}$ is denoted by $p(\boldsymbol{\xi})$, $R(\boldsymbol{\xi})$ refers to the return of an episode $\boldsymbol{\xi}$ and $\boldsymbol{\Xi}$ is the set of all possible paths. Using the Markov assumption, we can write the path distribution as $p(\boldsymbol{\xi}) = p(\mathbf{s}_1) \prod_{t=1}^{T+1} p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t) \pi(\mathbf{a}_t|\mathbf{s}_t, t)$ 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 $R(\boldsymbol{\xi}) = 1/T \sum_{t=1}^{T} r(\mathbf{s}_t, \mathbf{a}_t, \mathbf{s}_{t+1}, t)$, where $r(\mathbf{s}_t, \mathbf{a}_t, \mathbf{s}_{t+1}, t)$ denotes the immediate reward.

While episodic RL problems with finite horizons are common in motor control, few methods exist in the RL literature. Notable exceptions are model-free methods such as episodic REINFORCE [17] and the episodic natural actor-critic (eNAC) [4], as well as model-based methods, e.g., differential-dynamic programming [18]. To avoid learning of complex models, we focus on model-free methods. To reduce the number of open parameters, we use a novel RL algorithm called policy learning by weighting exploration with the returns (PoWER). This method is shown as pseudocode in Table 4. PoWER is an expectation-maximization algorithm where the action is treated similar as an unobserved variable and the returns are considered an improper probability distribution. When evaluating several methods in [16], we could show that PoWER clearly outperforms preceding policy search approaches like finite difference gradients (FDGs), vanilla policy gradients (VPGs), eNAC, and the episodic reward-weighted regression (eRWR) on two benchmark problems [4], [15]. See Table 5 for results on the underactuated swing-up benchmark [18]. Recently, PoWER has been extended and successfully applied in mobile robotics [19].

When learning MPs, we intend to learn a deterministic mean policy $\bar{\mathbf{a}} = \boldsymbol{\theta}^{\mathrm{T}} \boldsymbol{\phi}(\mathbf{s}) = \mathbf{f}(\mathbf{z})$ that has the basis functions $\boldsymbol{\phi}$, is linear in parameters $\boldsymbol{\theta}$ and is augmented by additive exploration $\boldsymbol{\varepsilon}(\mathbf{s}, t)$ in order to make model-free RL possible. As a result, the explorative policy can be given in the form $\mathbf{a} = \boldsymbol{\theta}^{\mathrm{T}} \boldsymbol{\phi}(\mathbf{s}, t) + \boldsymbol{\epsilon}(\boldsymbol{\phi}(\mathbf{s}, t))$. Previous work, with the notable exception of [20], has focused on state-independent, white Gaussian exploration [4], i.e., $\boldsymbol{\epsilon}(\boldsymbol{\phi}(\mathbf{s}, t)) \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma})$, and has resulted in applications such as T-ball batting [4] and constrained movement [7]. However, from our experience, such unstructured exploration at every step has several disadvantages, i.e., 1) it causes a large variance in parameter updates which grows with the number of time steps, 2) it perturbs actions too frequently, as the system acts as a low pass filter the perturbations average out and thus, their effects are washed out and 3) can damage the system executing the trajectory.

Alternatively, one could generate a form of structured, state-dependent exploration $\boldsymbol{\epsilon}(\boldsymbol{\phi}(\mathbf{s}, t)) = \boldsymbol{\varepsilon}_t^{\mathrm{T}} \boldsymbol{\phi}(\mathbf{s}, t)$ 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 [20]. Each σ_{ij}^2 corresponds to one θ_{ij}. This argument results in the policy $\mathbf{a} \sim \pi(\mathbf{a}_t|\mathbf{s}_t, t) = \mathcal{N}(\mathbf{a}|\boldsymbol{\phi}(\mathbf{s}, t), \hat{\boldsymbol{\Sigma}}(\mathbf{s}, t))$. This form of policies improves upon the shortcomings of

	FDG	VPG	eNAC	eRWR	PoWER
Final return	0.907	0.907	0.910	0.900	0.937
Episodes to $R = 0.9$	110	100	40	186	48

TABLE 5
Comparison of policy search algorithms on the underactuated swing-up [18] averaged over ten runs.

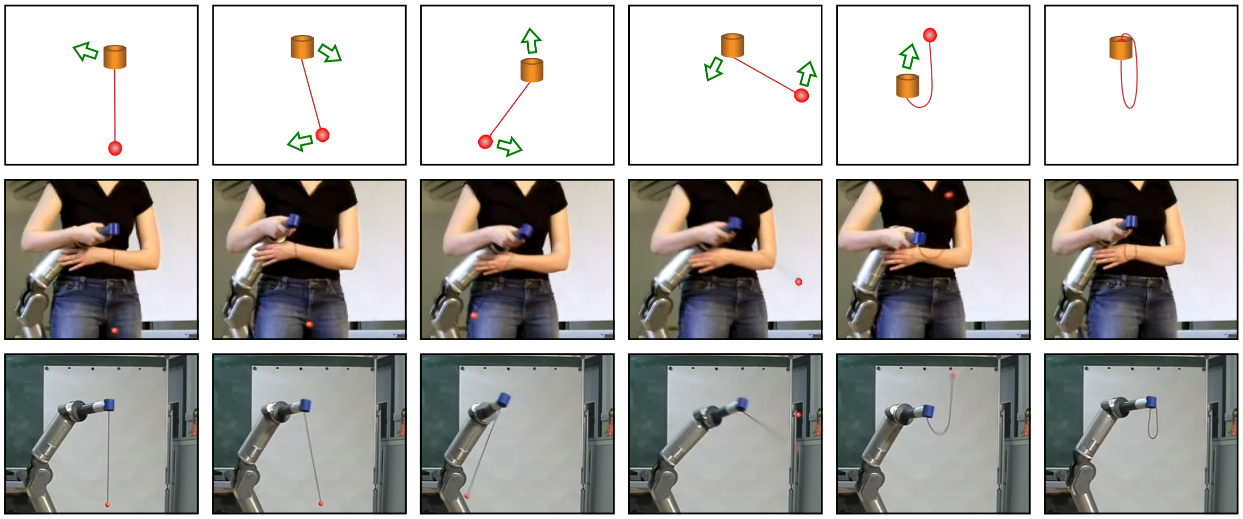
Fig. 2. The schematic drawings of (a) the ball-in-a-cup motion, (b) a kinesthetic teach-in, as well as (c) the performance of the robot after both imitation learning and RL.

directly perturbed policies mentioned earlier. Based on the EM updates for RL as suggested in [16], we get the update rule

$$\boldsymbol{\theta}' = \boldsymbol{\theta} + \frac{E_{\boldsymbol{\tau}}\left\{\sum_{t=1}^{T} \boldsymbol{\varepsilon}_t Q^{\pi}(\mathbf{s}_t, \mathbf{a}_t, t)\right\}}{E_{\boldsymbol{\tau}}\left\{\sum_{t=1}^{T} Q^{\pi}(\mathbf{s}_t, \mathbf{a}_t, t)\right\}} \tag{10}$$

for dynamical systems MPs, 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 metaparameter. Alternative state-based exploration methods have been suggested previously in [21] and [4].

To reduce the number of trials in this on-policy scenario, we reuse the trials through importance sampling [13]. To avoid the fragility sometimes resulting from importance sampling in RL, samples with very small importance weights are discarded.

PoWER is based on the idea of reward-weighted imitation and relies on the following intuition: a safe way to generate new policies is to look in the convex combination of sampled policies. The algorithm searches for a good solution between previously seen ones by staying close to policies with high returns and far from solutions with low returns. PoWER is a local learning method and is only guaranteed to converge to a local optimum. Thus, the method is prone to fail if the initial demonstration is too far from a good solution.

The more shape parameters $\boldsymbol{\theta}$ are used, the more details can be captured in a MP and it can ease the imitation learning process. However, if the MP need to be refined by RL, each additional parameter slows down the learning process. For optimal performance, we need a good initial value of σ_{ij}^2, which may be seen as a safety-speed tradeoff parameter. The parameter determines the exploration, where larger values lead to greater changes in the mean policy and, thus, may lead to faster convergence. However, they can also drive the robot into unsafe regimes. For a small value, convergence to a good solution will take a long time. As the method is based on reward-weighted imitation the exploration has to be sufficiently rich to avoid getting stuck in local minima. The optimization of the parameters decreases the exploration during the learning process and results in a convergence to a deterministic solution.

Robot Evaluation

The methods presented in this paper are evaluated on two learning problems on a real robot, i.e., we learn the discrete task ball-in-a-cup and the rhythmic task ball paddling. The resulting simplicity and speed of the learning process demonstrate the applicability of the MP-based learning framework for practical applications.

Our experimental setup includes a backdriveable 7 DoF Barrett WAM that can be used as a haptic input device for imitation and an in-house-developed vision system for ball tracking. The vision system consists of a stereo camera setup with two Prosilica GE640C Gigabit Ethernet cameras and a 200-Hz blob detection implemented on a NVIDIA graphics card.

Discrete Movement: Ball-in-a-Cup

The children's motor skill game ball-in-a-cup, also known as balero and bilboquet, is challenging even for an adult. The toy has a small cup which is held in one hand or, in our case, is attached to the endeffector of the robot. The cup has a small ball hanging down on a string where the string has a length of 40 cm for our toy. Initially, the ball is hanging down vertically at the 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

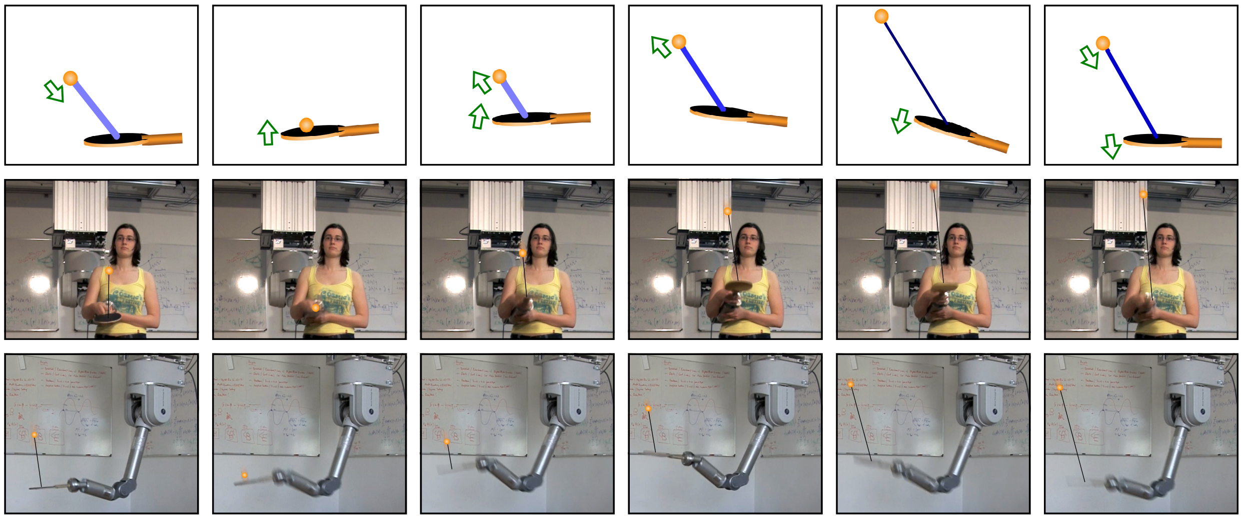

Fig. 4. The schematic drawings of (a) the ball-paddling motion, (b) a kinesthetic teach-in as well as (c) the performance of the robot after imitation learning.

illustrated in Figure 2(a). The human demonstration was taught to the robot by imitation learning with 31 parameters per joint for an approximately 3-s long trajectory [Figure 2(b)]. 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 PoWER algorithm from Table 4, 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 [Figure 2(c)].

Note that learning ball-in-a-cup and kendama have previously been studied in robotics and we are going to contrast a few of these approaches here. While we learn directly in the joint space of the robot, Takenaka [22] recorded planar human cup movements and determined the required joint movements for a planar, 3 DoF robot so that it could follow the trajectories while visual feedback was used for error compensation. Both Sato et al. [23] and Shone et al. [24] used motion-planning approaches that relied on very accurate models of the ball while employing only 1 DoF in [24] or 2 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. Probably, the most advanced preceding work on learning kendama was done by Miyamoto et al. [25] who used a 7 DoF anthropomorphic arm and recorded human motions to train a neural network to reconstruct via points. Employing full kinematic knowledge, the authors optimized a desired trajectory represented by splines.

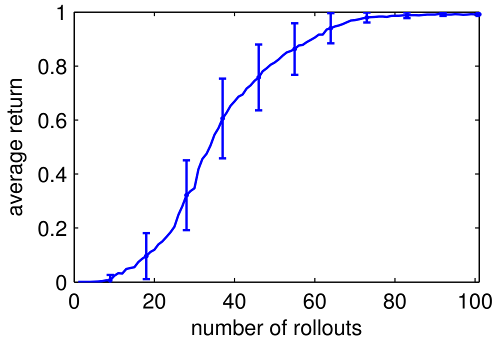

Fig. 3. The expected return of the learned policy in the ball-in-a-cup evaluation averaged over 20 runs.

The state of the system is 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 MP with one common canonical system. The movement uses all 7 DoFs and is not restricted to being planar. All MPs are perturbed separately but receive the same shared 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 $t \neq t_c$. 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 significantly and hence the outcome of the trial. The position of the ball is estimated using a stereo vision system and needed to determine the reward.

Because of the complexity of the task, ball-in-a-cup is a difficult motor task for children; they usually only succeed after observing another person presenting a demonstration first and through subsequent trial-and-error-based learning. Mimicking how children learn to play ball-in-a-cup, we first initialize the MPs by imitation and, subsequently, improve them by RL.

We recorded the motions of a human player by kinesthetic teach-in to obtain an example for imitation as shown in Figure 2 (b). Kinesthetic teach-in (i.e., taking the robot by the hand) requires 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 multiple demonstrations did not improve the performance, as the task is sensitive to small deviations. As expected, the robot fails to reproduce the presented behavior when learned purely by imitation. Thus, RL is needed for self-improvement of the performance. The imitation serves the learning system with a policy that swings the ball roughly to the height of the rim so that it may get a small reward based on the proximity to the rim.

The more parameters that are used for RL, the slower the convergence. Hence, prior to starting to learn by RL, we have determined that only 31 shape-parameters per MP suffice by cross-validation for the imitation learning-based initialization. Our best performing algorithm in previous comparisons, PoWER, was employed for all RL evaluations. This algorithm is also shown in Table 4. Here, the metaparameters σ_{ij} are initially set in the order of magnitude of the median of the parameters for each MP and are then optimized alongside the shape parameters. The performance of the algorithm is fairly robust for values chosen in this range.

Figure 3 shows the expected return over the number of rollouts with clear convergence to a maximum. The robot regularly succeeds at bringing the ball into the cup after approximately 75 rollouts. After 100 rollouts, the metaparameters, such as the exploration rate, have converged to negligible size and do not influence the outcome of the behavior any longer.

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 as children naturally get tired or bored.

Rhythmic Movements with Start-Up: 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 the ball falling down but also pulls the ball back towards the center of the paddle if the ball is hit sufficiently hard, i.e., the elastic string is also stretched as a consequence. The task is fairly easy to perform 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 to obtain a demonstration for imitation learning, as shown in Figure 4. When the string is stretched it is shown as thinner and darker. The human demonstration was taught to the robot by imitation learning. From the imitation, it was determined by cross-validation that 10 shape-parameters per MP are sufficient. The shape parameters, the amplitude, and the period of the motion are estimated from the demonstration after the start-up phase. An additional discrete MP is used for the start-up phase (see the accompanying video for details).

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 ball 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; this was the case not only for the robot but for all human presenters. For our setup, the start-up phase consists of moving the paddle slower and further up than during the rhythmic behavior, as exhibited by the teacher's movements. This kind of movement can easily be achieved in the dynamical systems MP framework by imposing another discrete dynamical systems primitive that gradually adapts the period parameter τ and the transformation function $\mathbf{f}$ to the ones encountered in the rhythmic behavior, as presented in Table 3. The discrete modifier MP 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 T_{start}. With this start-up phase, imitation learning from demonstrations suffices to reproduce the motor skill successfully. Previous work [27] considered the superposition of discrete and rhythmic primitives as well as modulating the offset of a rhythmic primitive by a discrete one. To our knowledge, this application is among the first where a rhythmic dynamical systems primitive is modified by a discrete primitive in the start-up phase to achieve a particular task.

Conclusion

In this article, we present both novel learning algorithms and experiments using the dynamical systems MPs. As such, we describe this MP representation in a way that it is straightforward to reproduce. We review an appropriate imitation learning method, i.e., locally weighted regression, and show how this method can be used both for initializing RL tasks as well as for modifying the start-up phase in a rhythmic task. We also show our currently best-suited RL algorithm for this framework, i.e., PoWER. We present two complex motor tasks, i.e., ball-in-a-cup and ball paddling, learned on a real, physical Barrett WAM, using the methods presented in this article. Of particular interest is the ball paddling application, as it requires the combination of both rhythmic and discrete dynamical systems MPs during the start-up phase in order to achieve a particular task.

Keywords

imitation learning, reinforcement learning, motor primitives, motor skill learning

References

[1] T. Flash and B. Hochner, "Motor primitives in vertebrates and invertebrates," *Current Opinions in Neurobiology*, vol. 15, no. 6, pp. 660–666, 2005.

[2] A. J. Ijspeert, J. Nakanishi, and S. Schaal, "Movement imitation with nonlinear dynamical systems in humanoid robots," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, 2002, pp. 1398–1403.

[3] J. Kober, B. Mohler, and J. Peters, "Learning perceptual coupling for motor primitives," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Syst. (IROS)*, 2008, pp. 834–839.

[4] J. Peters and S. Schaal, "Reinforcement learning of motor skills with policy gradients," *Neural Networks*, vol. 21, no. 4, pp. 682–697, 2008.

[5] D. Pongas, A. Billard, and S. Schaal, "Rapid synchronization and accurate phase-locking of rhythmic motor primitives," in *Proc. IEEE/RAS Int. Conf. Intelligent Robots and Syst. (IROS)*, 2005, pp. 2911–2916.

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

[7] 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.

[8] H. Urbanek, A. Albu-Schäffer, and P. van der Smagt, "Learning from demonstration repetitive movements for autonomous service robotics," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Syst. (IROS)*, 2004, pp. 3495–3500.

[9] S. Schaal, A. Ijspeert, and A. Billard, "Computational approaches to motor learning by imitation," *Philosph. Trans. Roy. Soc. London: Series B, Biological Sci.*, vol. 358, no. 1431, pp. 537–547, 2003.

[10] P. Abbeel and A. Y. Ng, "Apprenticeship learning via inverse reinforcement learning," in *Proc. 21st Int. Conf. Mach. Learning (ICML)*, 2004, p. 1.

[11] 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.

[12] D.-H. Park, H. Hoffmann, P. Pastor, and S. Schaal, "Movement reproduction and obstacle avoidance with dynamic movement primitives and potential fields," in *Proc. IEEE Int. Conf. Humanoid Robots (HUMANOIDS)*, 2008, pp. 91–98.

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

[14] M. Riedmiller, T. Gabel, R. Hafner, and S. Lange, "Reinforcement learning for robot soccer," *Autonomous Robots*, vol. 27, no. 1, pp. 55–73, 2009.

[15] J. Peters and S. Schaal, "Learning to control in operational space," *Int. J. Robotics Research*, vol. 27, no. 2, pp. 197–212, 2008.

[16] J. Kober and J. Peters, "Policy search for motor primitives in robotics," in *Advances Neural Inform. Process. Syst. (NIPS)*, D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, Eds. MIT Press, 2009, pp. 849–856.

[17] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Mach. Learning*, vol. 8, no. 3-4, pp. 229–256, 1992.

[18] C. G. Atkeson, "Using local trajectory optimizers to speed up global optimization in dynamic programming," in *Advances Neural Inform. Process. Syst. (NIPS)*, J. D. Cowan, G. Tesauro, and J. Alspector, Eds., 1994, pp. 503–521.

[19] N. Vlassis, M. Toussaint, G. Kontes, and S. Piperidis, "Learning model-free robot control by a monte carlo em algorithm," *Autonomous Robots*, vol. 27, no. 2, pp. 123–130, 2009.

[20] T. Rückstieß, M. Felder, and J. Schmidhuber, "State-dependent exploration for policy gradient methods," in *Proc. European Conf. Mach. Learning (ECML)*, 2008, pp. 234–249.

[21] R. Dearden, N. Friedman, and S. Russell, "Bayesian q-learning," in *Conf. Artificial Intell. / Innovative Applicat. Artificial Intell. Conf. (AAAI/IAAI)*, 1998, pp. 761–768.

[22] K. Takenaka, "Dynamical control of manipulator with vision: "cup and ball" game demonstrated by robot," *Trans. Japan Soc. Mech. 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," *Trans. Japan Soc. Mech. Engineers. C*, vol. 59, no. 558, pp. 487–493, 1993.

[24] T. Shone and G. Krudysz, "Dynamic manipulation of kendama," Research Project, Rensselaer Polytechnic Inst., Troy, NY, USA, 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 Neural Inform. Process. Syst. (NIPS)*, D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, Eds. MIT Press, 2009, pp. 297–304.

[27] S. Degallier, L. Righetti, L. Natale, F. Nori, G. Metta, and A. Ijspeert, "A modular bio-inspired architecture for movement generation for the infant-like robot icub," in *Proc. 2nd IEEE RAS/EMBS Int. Conf. Biomedical Robotics and Biomechatronics (BioRob)*, 2008, pp. 795 – 800.

Jens Kober is a Ph.D. student at the Robot Learning Lab at the Max-Planck Institute for Biological Cybernetics (MPI). He is a Student Member of the IEEE. His research interests include robotics, control, and machine learning.

Jan Peters heads the Robot Learning Laboratory at the MPI while being an invited researcher at the Computational Learning and Motor Control Laboratory at the University of Southern California (USC). Before joining MPI, he graduated from USC with a Ph.D. degree in computer science. He is a Member of the IEEE. His research interests include robotics, nonlinear control, machine learning, RL, and motor skill learning.