Auton Robot (2018) 42:529–551
https://doi.org/10.1007/s10514-017-9648-7

Using probabilistic movement primitives in robotics

Alexandros Paraschos[1] · **Christian Daniel**[2] · **Jan Peters**[1,3] · **Gerhard Neumann**[4]

Received: 15 December 2015 / Accepted: 23 June 2017 / Published online: 15 July 2017

Abstract Movement Primitives are a well-established paradigm for modular movement representation and generation. They provide a data-driven representation of movements and support generalization to novel situations, temporal modulation, sequencing of primitives and controllers for executing the primitive on physical systems. However, while many MP frameworks exhibit some of these properties, there is a need for a unified framework that implements all of them in a principled way. In this paper, we show that this goal can be achieved by using a probabilistic representation. Our approach models trajectory distributions learned from stochastic movements. Probabilistic operations, such as conditioning can be used to achieve generalization to novel situations or to combine and blend movements in a principled way. We derive a stochastic feedback controller that reproduces the encoded variability of the movement and the coupling of the degrees of freedom of the robot. We evaluate and compare our approach on several simulated and real robot scenarios.

Keywords Imitation learning · Movement primitives · Trajectory representation · Control · Robotics

✉ Alexandros Paraschos
Paraschos@ias.tu-darmstadt.de

Christian Daniel
Christian.Daniel@de.bosch.com

Jan Peters
Peters@ias.tu-darmstadt.de

Gerhard Neumann
Neumann@ias.tu-darmstadt.de

[1] Technische Universität Darmstadt, Hochschulstrasse 10, 64289 Darmstadt, Germany

[2] Bosch Center for Artificial Intelligence, Robert-Bosch-Campus, 71272 Renningen, Germany

[3] Max-Planck-Institut für Intelligente Systeme, Spemannstrasse 38, 72076 Tübingen, Germany

[4] Computational Learning for Autonomous Systems, School of Computer Science, University of Lincoln, Brayford Pool, LN6 7TS Lincoln, UK

1 Introduction

Movement Primitives (MPs) are a well-established approach for representing movement policies in robotics. MPs have several beneficial properties; generalization to new situations, temporal modulation of the movement, co-activation of multiple primitives to concurrently solve multiple tasks, sequencing of primitives to generate longer and more complex movements, and they are easy to learn from demonstrations. Using such properties, MPs were successfully applied to reaching (dAvella and Bizzi 2005), locomotion (Dominici et al. 2011; Moro et al. 2012) and are state of the art for robot movement representation and generation. However, many approaches for movement generation based on MPs (Ijspeert et al. 2003; Williams et al. 2007; dAvella and Bizzi 2005; Khansari-Zadeh and Billard 2011; Rozo et al. 2013; Rückert et al. 2012; Righetti and Ijspeert 2006) exhibit only a subset of these properties. Hence, a generalized framework that unifies all these properties in one principled framework is needed.

We formalize the concept of probabilistic movement primitives (ProMPs) as a general probabilistic framework for representing and learning MPs. A ProMP represents a distribution over trajectories. The trajectory distribution can be defined in either joint-space, task-space, or any other space that accommodates the experiment. In this paper, we focus on joint-space trajectories. Working with distributions enables us to formulate the described properties using operations

from probability theory. For example, modulation of a movement to a novel target can be realized by conditioning on the desired target's positions or velocities. Similarly, consistent parallel activation of two elementary behaviors can be accomplished by a product of two independent trajectory distributions. A trajectory distribution can encode the variance of the movement, and, hence, a ProMP can directly encode optimal behavior in systems with linear dynamics, quadratic costs and Gaussian noise (Todorov and Jordan 2002). In contrast, deterministic approaches, e.g., the DMP approach, can only represent the mean solution, which is known to be suboptimal. Even if assumption does not hold, we believe that it offers a good a approximation of physical robotic systems. Finally, a probabilistic framework allows us to model the coupling between the degrees of freedom (DoFs) of the robot by estimating the covariance between different DoFs.

The benefits of using a probabilistic representation have so far not been extensively exploited for representing and learning MPs. The main reason for this limitation has been the difficulty of extracting a policy for controlling the robot from a trajectory distribution. We show how this step can be accomplished and derive a control policy that exactly reproduces a given trajectory distribution. While ProMP introduces many novel components, it also incorporates many of the advantages from well-known previous movement primitive representations (Schaal et al. 2005; dAvella and Bizzi 2005), such as temporal rescaling of movements and the ability to represent both rhythmic and stroke based movements.

In this paper, we unify and complement our prior work (Paraschos et al. 2013a, b; Neumann et al. 2014) on ProMPs. Note that the reference Neumann et al. (2014) contains only a brief summary of our work on ProMPs presented in the context of an overview paper that spans over multiple topics. Therefore, Neumann et al. (2014) provides less information than the corresponding conference papers. In this paper, we present much more details which are necessary to reproduce the results. We introduce a new regularization technique for achieving smoother movements and present an expectation-maximization algorithm for learning rhythmic ProMPs in more detail. We extended the description of our controller derivation and show how it is used on physical tasks, e.g. controlling a 7-DoF arm for playing Maracas, robot-hockey, and 'Astrojax'. Moreover, we show new comparisons to state of the art MP approaches in terms of optimality, generalizability, composition of primitives and robustness of the movement representations. We also evaluate our ProMP controller on non-linear systems and made the source code of all examples publicly available.[1]

[1] http://www.ausy.tu-darmstadt.de/uploads/Team/AlexandrosParaschos/ProMP_toolbox.zip.

2 Properties of movement primitive frameworks

We categorize MPs into state-based (Khansari-Zadeh and Billard 2011; Calinon et al. 2010a) and trajectory-based representations (Schaal et al. 2005; Neumann et al. 2009; Rückert et al. 2012; Rozo et al. 2013). Trajectory-based primitives typically use time as the driving force of the movement. They require simple, typically linear, controllers, and scale well to a large number of DoFs. In contrast, state-based primitives (Khansari-Zadeh and Billard 2011; Calinon et al. 2010a) do not require the knowledge of a time step but often need to use more complex, non-linear policies. Such increased complexity has limited the application of state-based primitives to a rather small number of dimensions, such as the Cartesian coordinates of the task space of a robot. The main focus of this paper is on trajectory-based representations. We begin with a discussion on the properties of MPs.

2.1 Concise representation

MPs offer a concise representation of the movement, with a few open parameters to set. The small number of parameters simplifies learning the movement from demonstrations and the use of reinforcement learning algorithms to adapt and refine the primitive through trial-and-error. MP frameworks can be trained from demonstrations using simple learning methods, e.g. linear regression, and have been successfully used in fairly complex scenarios, including "Ball-in-the-Cup" Kober et al. (2010), Ball-Throwing (Ude et al. 2010; da Silva et al. 2012), Pancake-Flipping (Kormushev et al. 2010), Tetherball (Daniel et al. 2012a), and bi-pedal locomotion (Nakanishi et al. 2004).

2.2 Adaptation and time modulation

Many MPs offer an intrinsic adaptation mechanism to match a new situation or an altered task, e.g., hitting a different incoming balls when playing table tennis. The adaptation commonly comes in a form of modification of the desired target position and velocity at the end of the primitive or as a modulation of the amplitude of the primitive (Ijspeert et al. 2003). Our approach (Paraschos et al. 2013a, b) can be used to adapt the movement at any time point during the trajectory's execution.

Furthermore, adaptation of MPs include temporal modulation. Temporal modulation is a valuable property as it enables MPs to be applied in scenarios where correct timing is critical for the success of the task, e.g., in hitting, batting, or in locomotion to adjust the walking speed of the robot (Righetti and Ijspeert 2006).

2.3 Combination and sequencing

The expressiveness of an MP approach can be significantly improved if multiple primitives can be simultaneously co-activated to compose more complex movements. However, most MP approaches do not support co-activation of primitives in a principled way. Instead, the concurrent activation requires a prioritization scheme (Mülling et al. 2013; Pastor et al. 2011) in order not to disrupt the motion. In our approach (Paraschos et al. 2013a), we co-activate primitives to solve multiple tasks at the same time, without the need of such a scheme. Besides simultaneous activation, MP architectures aim to support sequencing MPs (Konidaris et al. 2012) to acquire a smooth transition from one primitive to another. Such sequencing is needed to dynamically concatenate primitives in order to acquire longer, more complex movements. We show that in our framework a smooth transition can be achieved in a principled way similar to the combination of primitives.

2.4 Coupling the DoFs

Movement primitives approaches are typically applied to robots with multiple Degrees of Freedom (DoF). In order to reproduce coordinated movements, MPs need a synchronization mechanism among the different DoF. Using time, or a function of time, as a reference signal (Schaal et al. 2007; Ijspeert 2008), one can implement simple time alignment mechanisms. However, when experiencing deviations from the desired trajectory due to noise or unmodeled effects, coordinated recovering from perturbations is advantageous. ProMPs, additionally to time synchronization, estimate such correlations directly from demonstrations and use them to synchronize the DoFs of the system.

2.5 Optimal behavior

Many trajectory-based representations use a single desired trajectory that is followed by a feedback controller with constant gains. However, following such a single trajectory has been proven to be suboptimal for many tasks if the system's dynamics are stochastic (Todorov and Jordan 2002). In this paper, we focus on control affine systems with Gaussian control noise, which is a standard assumption for physical systems. In this case, a distribution over trajectories is a good representation of the optimal behavior. Such distribution can be achieved by using time-varying feedback gains, which are often used as approximation for optimal behavior (Li and Todorov 2010). Feedback controllers with time varying gains modulate the stiffness of the system to provide high precision at the 'important' time points of a task while the system is less controlled for time points where accurate control in not so critical. The time varying gains of the controller can be approximated (Calinon et al. 2010b), computed using a LQR by specifying a cost function (Calinon 2016; Bruno et al. 2015), improved with reinforcement learning (Buchli et al. 2011), or, as in our approach, computed in closed form (Paraschos et al. 2013a).

2.6 Stability

Generating stable behavior is an important aspect of MPs. However, stability guaranties often have limited use as they assume linearity in the dynamics. Yet, however simple, real-world, systems are non-linear, e.g., a pendulum, where the gravity alone introduces non-linearities in the dynamics. Discrete DMPs (Ijspeert et al. 2003) generate stable behavior by moving towards an attractor at the end of the movement, while periodic MPs (Ijspeert et al. 2003; Righetti and Ijspeert 2006) stabilise the movement on a unit circle. The probabilistic framework from Calinon et al. (2010a) initially did not provide any stability guarantees, but it was still generating stable movements as long as the disturbances did not perturb the system "far" from the region where the demonstration occurred. With Khansari-Zadeh and Billard (2011) the authors alleviate the problem and learned asymptotically stable control laws. Recently, Calinon (2016) proposed the use of a Linear Quadratic Regulator (LQR) for control, that is stable for closed-loop systems (Stengel 2012). The ProMPs (Paraschos et al. 2013a) derive a controller that exactly reproduces the demonstrated trajectory distribution and, thus, provide stability guaranties as long as the demonstrated trajectory distribution was generated by a stable control law.

3 Related work

A commonly used trajectory-based representation is the Dynamic Movement Primitive (DMP) approach, introduced in Ijspeert et al. (2003, 2013) for a recent review. They represent a linear attractor system which is modulated by a time-dependent forcing function. The DMP introduced the concept of a phase, defined as a monotonic function of time. By adjusting the phase derivatives, we can temporally scale the movement. The forcing function is represented by normalized Gaussian basis functions, multiplied with the phase signal. Since the phase decreases exponentially to zero, the forcing function will asymptotically vanish at the end of the movement. At that time, only the attractor dynamics stay active, which guarantees the stability of the linear system. When used in an imitation learning scenario, the weights of the basis functions can be fitted from a single demonstration using linear regression. Generalization to new, unseen, situations in DMPs is limited. The original formulation only allowed for changing the position at the end of the move-

ment, which is implemented by modifying the position of the goal attractor or, for rhythmic DMPs, by adjusting the amplitude of the forcing function. Extensions exist that also allow setting a desired final velocity (Kober et al. 2010; Mülling et al. 2013; Paraschos et al. 2009). Directly changing intermediate points in the trajectory is not possible. DMPs can be sequenced given proper initialization (Paraschos et al. 2009), but only instant switching from one primitive to another is considered. Kulvicius et al. (2012) extended DMPs to support sequencing of primitives and evaluated their approach on a handwriting dataset. Gams et al. (2014) proposed the use of DMPs for tasks that include interactions with the environment.

Despite that DMPs introduce many beneficial properties, such as temporal scaling of the movement, learning from a single demonstration or generalizing to new final positions, further work is still needed for concurrently activating multiple primitives, generalizing to intermediate via-points, representing optimal behavior in stochastic systems, and capturing the correlation of the individual joints of the robot. Trajectories based on DMPs applied to multiple DoF systems are synchronized based only on the internal phase variable. Multiple DMPs for the same DoF cannot be activated simultaneously without further considerations on prioritized control and partial cancellation of the movement.

Probabilistic approaches use distributions to additionally encode the variability of the movement (Calinon et al. 2010a; Rozo et al. 2013; Kormushev et al. 2010; Calinon 2016). The variability of the movement, or the variance in distribution terms, is crucial, as it reflects the importance of single time points for the movement execution and it is often a requirement for representing optimal behavior in stochastic systems (Todorov and Jordan 2002). Moreover, capturing the variance of the movement leads to better generalization capabilities and to more natural movements. A probabilistic MP approach was proposed by Calinon et al. (2010a), where a Gaussian Mixture Regression (GMR) model was used to represent the trajectory. Given a set of trajectories, the GMR was trained with an Expectation Maximization (EM) algorithm (Rozo et al. 2013). A unifying formulation that extends the DMPs and uses them in a probabilistic framework is discussed in Kormushev et al. (2010). Yet, it is unclear how a GMR model can be conditioned to reach different final or intermediate positions. An extension of the approach (Calinon 2016) enabled generalization to different situations by recording the movement from different spaces and tracking the affine transformation to each space. While the approach is capable of generalizing, for example when an object changes its position, it can not modulate the encoded variance.

Besides representing the variance of the trajectory, we need a controller that reproduces the encoded distribution on a real system. A feedback controller where the gains are based on the inverse of the covariance of the current time-step was presented in Calinon et al. (2010b). The control law is based on the intuition that the gains have to be lower when the variance of the trajectories is higher. A comparison to this control law is presented at the evaluation section of this paper. As our experiments show, the resulting trajectory distribution from executing this controller does not match the desired one. In Calinon (2016) and Bruno et al. (2015), the authors proposed the use of minimum intervention control to generate the gains of the feedback controller. In this approach, the authors use the inverse of the covariance at every time point as metric for the quadratic state costs. However, while intuition-wise weighting the state with the inverse of the covariance is appropriate, we will show in our comparison that this approach can not match the desired trajectory distribution. Additionally, the cost function proposed by the authors include a quadratic action penalty to limit the actions that is not learned by the demonstrations.

A different approach for computing a control law for a GMR model was proposed by Khansari-Zadeh and Billard (2011). In this approach, the control gains are proven to be stable if the system is linear. The authors derive the stability constraints from the Lyapunov stability theory. In Khansari-Zadeh et al. (2014), the authors extend their approach to generate stable controllers with state-dependent stiffness. The resulting controller share similarities with the ProMPs controller.

The approach by Rückert et al. (2012) also offers a probabilistic interpretation of MPs by representing them with learned graphical models. A probabilistic planning algorithm is used to obtain a controller that optimizes the cost function represented by the graphical model. The resulting controller is also a linear feedback controller with time varying gains. However, this approach heavily depends on the quality of the used planner and imitation learning of such a representation is not straightforward.

The ability to combine multiple MPs into a single movement provides significantly better generalization capabilities, enables the use of MP libraries, and has recently attracted attention of the community. Mülling et al. (2013) use a library for table tennis which is concurrently activating multiple DMPs to perform striking movements. Each primitive is activated with an activation provided by a trained gating network. The primitives are then combined on the acceleration level which is equivalent to a linear combination of primitives in parameter space. The primitives and the activation weights were refined with Reinforcement Learning (RL). A different approach was proposed by Matsubara et al. (2011) using DMPs in combination of with a style parameter. The parameters of DMPs are linearly interpolated according to the given style parameter. Forte et al. (2012) proposed a similar approach, where a library of DMPs learned from multiple demonstrations is used. Generalization is obtained from a Gaussian Process Regression (GPR) model which

is capable of modeling non-linear transformations of the style variable. The major limitation of approaches based on deterministic representations, e.g., on DMPs, is the inability to concurrently solve a combination of tasks where we have one task per primitive. Since there is no notion of the importance of each time point in the trajectory the resulting combined primitive is just an interpolation of the participating primitives trajectories. In contrast, probabilistic representations (Khansari-Zadeh and Billard 2011; Calinon et al. 2010a) leave unclear how primitives can be combined. In ProMPs, we propose a new combination operator based on a product of trajectory distributions. We show that by co-activating ProMPs, the resulting movement solves a combination of tasks that is given by a combination of different cost functions. We evaluate this property in two different scenarios in the experiments section.

Smoothly sequencing, also called blending, two movement primitives can be considered as a special case of a combination of MPs. Discrete DMPs can be trivially sequenced (Paraschos et al. 2009), however the transition from one primitive to then next one is typically instantly, which can lead to a jump in the acceleration profiles. Special cases of discrete and periodic primitive blending, such as transient motions, have been considered in Ernesti et al. (2012) and Degallier et al. (2011). As opposed to the previous approaches, the ProMPs can cope with combination and blending of primitives independently of their periodicity.

In the next section, we will first introduce probabilistic movement primitives and show their advantageous properties. Next, will show how to compute a time-varying feedback controller that reproduces the given trajectory distribution. Subsequently, we will demonstrate the performance and advantageous properties of ProMPs in several experiments on simulated and real robot tasks.

4 Probabilistic movement primitives (ProMPs)

ProMPs provide a single principled framework for implementing the desirable properties of MPs, summarized in Table 1. We will first introduce the probabilistic model for representing the trajectory distribution, that is based on a basis function representation. Such a representations significantly reduces the amount of model parameters and facilitates learning. We proceed by illustrating how our representation can be trained from imitation data for both stroke-based and periodic movements. Training from imitation allows to rapidly reproduce tasks that are easy to demonstrate to the robot. Here, we describe a simple maximum likelihood training procedure that can be used for stroke-based movements and an expectation-maximization algorithm that can be used to train the primitive in case of missing data or also for rhythmic movements. We continue by discussing

Table 1 Properties and their implementation in the ProMPs

Property	Implementation
Co-activation	Product of $p_i(\boldsymbol{\tau})$
Modulation	Conditioning
→ final positions	✓
→ final velocities	✓
→ via-points	✓
Optimality	Encode variance
Coupling	Mean, covariance
Learning	Max. likelihood
Temporal modulation	Modulate phase
Rhythmic movements	Periodic basis

Fig. 1 Two real robot setups that we used for the evaluation of our approach. (*left*) The KUKA arm playing the maracas musical instrument. We demonstrated a slow version of the rhythmic shaking movement and we progressively increased the speed. (*right*) The KUKA arm playing with an Astrojax. The robot learned to play from demonstrations

the implementation of the desirable properties, i.e. temporal modulation of the movement, encoding of the coupling between the joints that allows the generation of coordinated movements, conditioning to generalize a trained primitive to a novel situation, adaptation to task parameters to allow task-dependent variables to modify the primitive, and combination and blending of primitives to solve more complex tasks. Finally, in Sect. 4.4, we present the analytical derivation of a stochastic feedback controller that is capable of exactly reproducing the trajectory distribution. Such feedback controller is essential for using trajectory distributions for controlling a physical system (Fig. 1).

4.1 Probabilistic trajectory representation

We start our discussion with the simple case of a single degree of freedom, where the joint angle q is a scalar, and we subsequently extend it to the multiple DoF case, where the vector $\boldsymbol{q}$ describes multiple joint angles. We model a single movement execution as a trajectory $\boldsymbol{\tau} = \{q_t\}_{t=0\ldots T}$, defined by the joint angle q_t over time. In our framework, a MP describes multi-

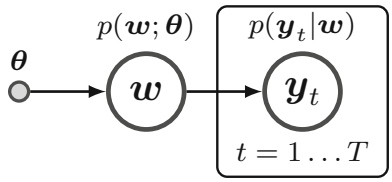


Fig. 2 The Hierarchical Bayesian model used in ProMPs. The probability distribution $p(\boldsymbol{y}_{1:T}|\boldsymbol{w})$ of the observed trajectories depends on the parameter vector $\boldsymbol{w}$. The distribution over the parameter vector $\boldsymbol{w}$ is given by $p(\boldsymbol{w}|\boldsymbol{\theta})$. The parameter vector $\boldsymbol{w}$ is integrated out in the ProMP formulation

ple ways to execute a movement, which naturally leads to a probability distribution over trajectories. We encode our policy representation with a hierarchical Bayesian model, which is presented in Fig. 2.

4.1.1 Concise encoding of trajectory distributions

Our movement primitive representation models the time-varying variance of the trajectories. Representing the variance information is crucial as it reflects the importance of single time points for the movement execution. We use a basis-function representation as it reduces the amount of model parameters in comparison to a simple distribution over the joint positions for each time step. This reduction in parameters can greatly facilitate learning. Additionally, it allows us to derive a continuous time approach and transfer data between systems, e.g., from a motion capture system to the robotic platform, directly without interpolating the data. When controlling the system, a continuous time approach allows for choosing the control frequency and is robust to jitter. Further, as we will discuss in Sect. 4.1.2, it enables the temporal modulation of the movement. Additionally, it allows us to generalize the primitive at any time-point, Sect. 4.3.1 and to derive our feedback controller in closed form, Sect. 4.4.

We use a weight vector $\boldsymbol{w}$ to compactly represent a single trajectory. The probability of observing a trajectory $\boldsymbol{\tau}$ given the underlying weight vector $\boldsymbol{w}$ is given as a linear basis function model

$$\boldsymbol{y}_t = \begin{bmatrix} q_t \\ \dot{q}_t \end{bmatrix} = \boldsymbol{\Phi}_t \boldsymbol{w} + \boldsymbol{\epsilon}_y, \tag{1}$$

$$p(\boldsymbol{\tau}|\boldsymbol{w}) = \prod_t \mathcal{N}\left(\boldsymbol{y}_t \middle| \boldsymbol{\Phi}_t \boldsymbol{w}, \boldsymbol{\Sigma}_y\right), \tag{2}$$

where $\boldsymbol{\Phi}_t = [\boldsymbol{\phi}_t, \dot{\boldsymbol{\phi}}_t]^{\mathrm{T}}$ defines the $2 \times n$ dimensional time-dependent basis function matrix for the joint positions q_t and velocities $\dot{q}_t$. The basis functions for the velocities $\dot{\boldsymbol{\phi}}_t$ are the time derivatives of $\boldsymbol{\phi}_t$. The variable n defines the number of basis functions and $\boldsymbol{\epsilon}_y \sim \mathcal{N}\left(\boldsymbol{0}, \boldsymbol{\Sigma}_y\right)$ represents zero-mean i.i.d. Gaussian noise.

In order to capture the variance of the trajectories, we introduce a distribution $p(\boldsymbol{w};\boldsymbol{\theta})$ over the weight vector $\boldsymbol{w}$, with parameters $\boldsymbol{\theta}$. In most cases, the distribution $p(\boldsymbol{w};\boldsymbol{\theta})$ will be Gaussian where the parameter vector $\boldsymbol{\theta} = \{\boldsymbol{\mu}_{\boldsymbol{w}}, \boldsymbol{\Sigma}_{\boldsymbol{w}}\}$ specifies the mean and the variance of $\boldsymbol{w}$. However, also more complex distributions such as Gaussian mixture models can be used for this task (Rueckert et al. 2015). The trajectory distribution $p(\boldsymbol{\tau};\boldsymbol{\theta})$ can now be computed by marginalizing out the weight vector $\boldsymbol{w}$, i.e.

$$p(\boldsymbol{\tau};\boldsymbol{\theta}) = \int p(\boldsymbol{\tau}|\boldsymbol{w}) p(\boldsymbol{w};\boldsymbol{\theta}) d\boldsymbol{w}, \tag{3}$$

to obtain the probability distribution over the trajectories $\boldsymbol{\tau}$. The distribution $p(\boldsymbol{\tau};\boldsymbol{\theta})$ defines the hierarchical Bayesian model that is illustrated at Fig. 2. The model's parameters are given by the observation noise variance $\boldsymbol{\Sigma}_y$ and the parameters $\boldsymbol{\theta}$ of the weight distribution $p(\boldsymbol{w};\boldsymbol{\theta})$.

Illustrative example To illustrate the properties of our MP representation, we use a simple toy-task as a running example throughout this section where we also compare to other state-of-the-art MP approaches. In our toy-task, we use a trajectory distribution that passes through two via-points. The simulated system has linear dynamics and Gaussian i.i.d. noise on the actions. In this illustrative example, we control the acceleration of the system. We generate demonstrations with an optimal control algorithm (Toussaint 2009). The cost function is given as

$$C(\boldsymbol{\tau}, \boldsymbol{u}) = \sum_{i=\{t_{\text{via}}\}} (\boldsymbol{y}_i^d - \boldsymbol{y}_i)^T \boldsymbol{Q} (\boldsymbol{y}_i^d - \boldsymbol{y}_i) + \sum_{i=1}^{T} \boldsymbol{u}_i^T \boldsymbol{R} \boldsymbol{u}_i, \tag{4}$$

where $t_{\text{via}} = \{0.4\,\text{s}, 0.7\,\text{s}\}$ is a set of the time-points for the via-points and $\boldsymbol{Q}$, $\boldsymbol{R}$ are are the state and action cost matrices, respectively. We simulate trajectories with the resulting controller to obtain the demonstrations. The demonstrations exhibit variability due to the noise of the system. The optimal trajectory distribution is presented in Fig. 3a.

The use of a cost-function enables us to quantify the quality of the resulting MP policies. The ProMP policy is capable of reproducing exactly the variance of the movement, as shown in Fig. 3b. For the trajectory reproduction of ProMPs, we used the controller that we describe in Sect. 4.4. Additionally, we evaluate the heuristic controller presented in Calinon et al. (2010b), which computes the feedback gains inverse proportionally to the variance of the trajectory. The trajectory distribution of the inverse covariance controller does not match the demonstrated distribution, see Fig. 3c. The DMP approach uses constant feedback gains to follow a single trajectory, and, hence, can not adapt the variance of the resulting trajectory distribution. In Fig. 3d, we generated trajectory distributions for two different settings of the feedback gains to illustrate the resulting variances. We empirically optimized

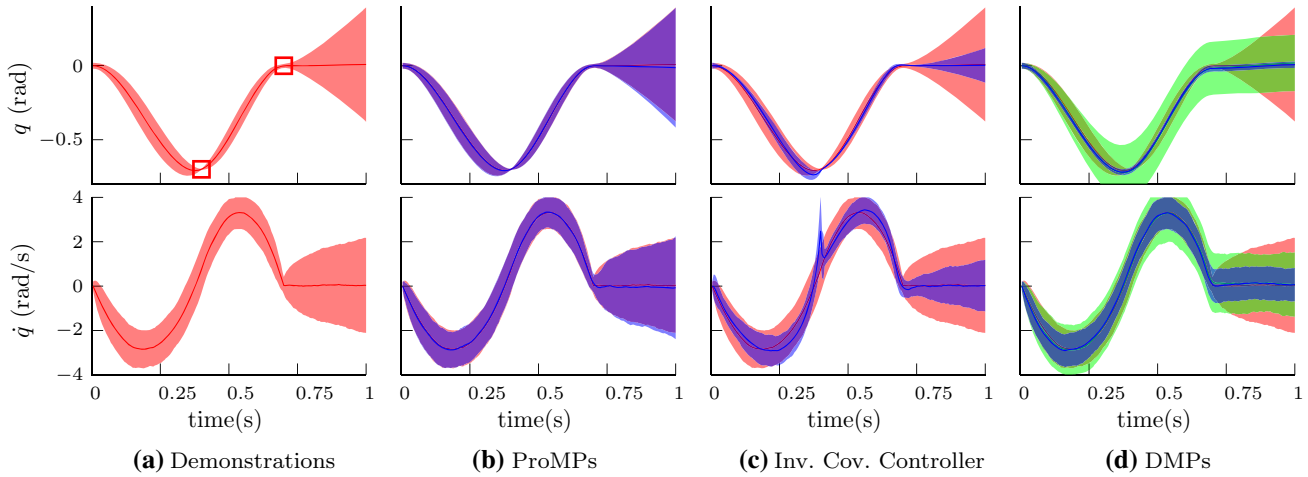


Fig. 3 Trajectory distribution showing the joint positions (*first row*) and velocities (*second row*). The *shaded area* denotes two times the standard deviation. **a** The demonstrated trajectory distribution that was generated by an stochastic optimal control algorithm for a via-point task. The resulting trajectories show variability due to the noise in the system. **b** The trajectory distribution generated using ProMPs (*blue*). ProMPs can exactly reproduce the demonstrated trajectory distribution (shown in *red* below the *blue shaded area*). **c** The resulting trajectory distribution produced by the inverse covariance control approach (*blue*). Due to latency-effects it missed the via-points in time and generated high actions which led to the velocity spike. **d** Trajectory distribution produced by DMPs. While the DMP can follow the mean of the demonstrations, it can not adapt its variance. The accuracy at the via-points is worse than ProMPs, while the control actions are higher in non-relevant areas of the trajectory. In *blue* we tuned the DMP gains for reproducing the trajectory distribution with the lowest cost and in *green* we used lower gains (Color figure online)

Table 2 Comparison of different control approaches on a hand-specified cost function

Control approach	Average cost
Reproduction	
Optimal controller	$2.07 \times 10^4 \pm 2.58 \times 10^2$
Model-free Gaus. Ctl	$2.25 \times 10^4 \pm 3.21 \times 10^2$
ProMP Jerk Penalty	$2.29 \times 10^4 \pm 3.35 \times 10^2$
ProMP weight reg.	$2.35 \times 10^4 \pm 3.25 \times 10^2$
Opt. Ctl.—Gaus. dist.	$3.37 \times 10^4 \pm 4.41 \times 10^2$
GMM/GMR—min int.	$4.47 \times 10^4 \pm 7.25 \times 10^2$
DMP	$5.16 \times 10^4 \pm 13.2 \times 10^2$
Inv. cov. controller	$7.36 \times 10^4 \pm 16.1 \times 10^2$
DMP with low gains	$76.5 \times 10^4 \pm 392 \times 10^2$
Combin.	
Optimal controller	$3.36 \times 10^4 \pm 3.52 \times 10^2$
ProMP	$5.46 \times 10^4 \pm 3.55 \times 10^2$
Inv. cov. controller	$6.54 \times 10^4 \pm 7.30 \times 10^2$
DMP	$208 \times 10^4 \pm 107 \times 10^2$

As baseline, we compare the approaches to an optimal controller that maximizes the cost. The ProMPs can produce trajectories with a similar cost. The newly presented regularization scheme for the weights (jerk penalty, Sect. 4.2.1) achieves a slightly lower costs due to the smoother torque profiles produced by this approach

the gains for the inverse covariance controller and the DMPs using search. The average costs generated by each control law are shown in the upper part of Table 2. The ProMP achieve a similar cost to the optimal controller while all other controllers can not reproduce the optimal behavior.

Further, we compare our approach to Calinon (2016), where we fit the proposed Gaussian Mixture Model (GMM) to the demonstrations and then use Gaussian Mixture Regression (GMR) to derive the desired trajectory distribution. We present the fitted regression model in Fig. 4 (blue). We generated trajectories using Minimum Intervention Control (Calinon 2016) and we present the results in Fig. 4 (red) where we jointly optimized for the number of mixture components and the action penalty. We also used the optimal number of components, but the same action penalty as in the cost function used to generate the demonstrations (green). The resulting controller can not reproduce the given distribution.

Moreover, we evaluated our approach using simple Gaussian distributions and optimal control. At every time-step, we fit a Gaussian distribution over the state and we use it to set a quadratic cost function. The cost function has the form of Eq. (4) where $\boldsymbol{y}_i^d$ is set to the mean and Q to the inverse of the covariance. We optimize for the action penalty $\boldsymbol{R}$ such that the true cost function we used to generate the data is minimized. We present our results in Table 2. This approach uses the same approach for deriving the controller as in Calinon (2016), but uses a simple Gaussian distribution to model each time-step instead of the state-defined GMR. Compared to ProMPs, the performance on the true cost function is worse as can be seen in the table. This approach also does not provide any generalization or modulation mechanism.

As another baseline, we fit a Gaussian distribution at every time-step on the state-action space. At reproduction, we condition the distribution of that time-step on the current state

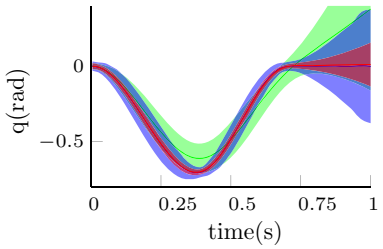


Fig. 4 Evaluation of the GMM-GMR approach, using the minimum innervation principle for control (Calinon 2016). The learned distribution using the GMM-GMR approach is presented in *blue*. The approach captures the mean of the distribution accurately, however, the variance at the via-points is higher than in the demonstrations. For reproduction, we used the optimal action penalty (*red*) or the same action penalty as in the demonstrations (*green*). While the mean of the reproductions matches the mean of the demonstrations, there is a miss-match for the variance (Color figure online)

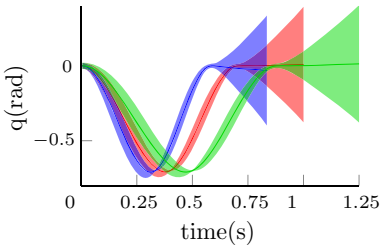


Fig. 5 Temporal modulation of the ProMPs. The demonstrated distribution is shown in *red*. The *green* shows an execution at a slower pace, whereas the *blue* at a faster one (Color figure online)

to obtain the action, which results in a linear Gaussian action policy. As the demonstrations have been generated by a time-dependent linear controller, the performance of this approach is is close to optimal and similar to the ProMP controller as shown in Table 2. However, fitting a Gaussian distribution over the state-action requires the actions to be known during the demonstrations and, which limits the applicability of the approach to tele-operation setups. Similar to the optimal control approach from the previous paragraph, this approach does not provide any generalization mechanism.

4.1.2 Temporal modulation

With temporal modulation, we can adjust the execution speed of the movement. Similar to the DMP approach, we introduce a phase variable z to decouple the movement from the time signal. By modifying the rate of the phase variable, we can modulate the speed of the movement. Without loss of generality, we define the phase as $z_0 = 0$ at the beginning of the movement and as $z_T = 1$ at the end. We typically use a constant velocity $\dot{z}_t = 1/T$ for reproducing the recorded motion, but we can also adapt it dynamically during the execution of the movement. The basis functions $\boldsymbol{\phi}_t$ now directly depend on the phase instead of time, such that

$$\boldsymbol{\phi}_t = \boldsymbol{\phi}(z_t), \tag{5}$$
$$\dot{\boldsymbol{\phi}}_t = \dot{\boldsymbol{\phi}}(z_t)\dot{z}_t, \tag{6}$$

where $\dot{\boldsymbol{\phi}}_t$ denotes the corresponding derivative. An illustration of temporal scaling for our running example is shown in Fig. 5.

4.1.3 Rhythmic and stroke-based movements

The choice of the basis functions depends on the type of movement, which can be either rhythmic or stroke-based. For stroke-based movements, we use Gaussian basis functions b_i^{G}, while for rhythmic movements, we use Von-Mises basis functions b_i^{VM} to model periodicity in the phase variable z, i.e.,

$$b_i^{\mathrm{G}}(z) = \exp\left(-\frac{(z_t - c_i)^2}{2h}\right), \tag{7}$$
$$b_i^{\mathrm{VM}}(z) = \exp\left(\frac{\cos(2\pi(z_t - c_i))}{h}\right), \tag{8}$$

where h defines the width of the basis and c_i the center for the ith basis function. We normalize the basis functions

$$\phi_i(z_t) = \frac{b_i(z)}{\sum_{j=1}^{n} b_j(z)}, \tag{9}$$

to obtain a constant summed activation and improve the regression's performance. The centers of the basis functions are uniformly placed in $[-2h, (1+2h)]$ the phase domain. We center basis functions outside the interval $[0, 1]$ to improve homogeneity of the basis vector, i.e., by including the "tails" of the basis placed outside, and therefore improve the performance of our model.

4.1.4 Encoding coupling between joints

So far, we have considered each degree of freedom to be modeled independently. However, for many tasks we have to coordinate the movement of multiple joints. The trajectory distributions $p(\boldsymbol{\tau}; \boldsymbol{\theta})$ can be easily extended to the multi-DoF case. For each dimension i, we maintain a parameter vector $\boldsymbol{w}_i$, and we define the combined weight vector $\boldsymbol{w}$ as $\boldsymbol{w} = [\boldsymbol{w}_1^T, \ldots, \boldsymbol{w}_n^T]^T$, a concatenation of the weight vectors. The basis matrix $\boldsymbol{\Phi}_t$ now extends to a block-diagonal matrix containing the basis functions and their derivatives for each dimension. The observation vector $\boldsymbol{y}_t$ consists of the angles and velocities of all joints. The probability of an observation $\boldsymbol{y}$ at time t is given by

Algorithm 1: Learning Stroke-Based Movements

Data: A set of N trajectories with position observations $\boldsymbol{Y}_i$, $i = 1 \ldots N$ at time $\boldsymbol{t}_i$.
Input: Number of basis functions K, Basis function width h, Regression parameter λ.
Result: The mean $\boldsymbol{\mu}_{\boldsymbol{w}}$ and covariance $\boldsymbol{\Sigma}_{\boldsymbol{w}}$ of $p(\boldsymbol{w}) \sim \mathcal{N}\left(\boldsymbol{w} \middle| \boldsymbol{\mu}_{\boldsymbol{w}}, \boldsymbol{\Sigma}_{\boldsymbol{w}}\right)$.
foreach *trajectory* i **do**
→ Compute phase: $z_i = {}^{t_i}/_{t_i^{\text{end}}}$;
→ Generate basis: $\boldsymbol{\Psi}_t = f(z_i, K, b)$, Eq. (9);
→ Compute the weight vector $\boldsymbol{w}_i$ for trajectory i

$$\boldsymbol{w}_i = \left(\boldsymbol{\Psi}_t^T \boldsymbol{\Psi}_t + \lambda \boldsymbol{I}\right)^{-1} \boldsymbol{\Psi}_t^T \boldsymbol{Y}_i.$$

end
→ Fit a Gaussian over the weight vectors w_i

$$\boldsymbol{\mu}_{\boldsymbol{w}} = \frac{1}{N}\sum_{i=1}^{N} \boldsymbol{w}_i, \quad \boldsymbol{\Sigma}_{\boldsymbol{w}} = \frac{1}{N}\sum_{i=1}^{N} (\boldsymbol{w}_i - \boldsymbol{\mu}_{\boldsymbol{w}})(\boldsymbol{w}_i - \boldsymbol{\mu}_{\boldsymbol{w}})^T.$$

return $\boldsymbol{\mu}_{\boldsymbol{w}}, \boldsymbol{\Sigma}_{\boldsymbol{w}}$.

$$p(\boldsymbol{y}_t|\boldsymbol{w}) = \mathcal{N}\left(\begin{bmatrix} \boldsymbol{y}_{1,t} \\ \vdots \\ \boldsymbol{y}_{d,t} \end{bmatrix} \middle| \begin{bmatrix} \boldsymbol{\Phi}_t & \ldots & \boldsymbol{0} \\ \vdots & \ddots & \vdots \\ \boldsymbol{0} & \ldots & \boldsymbol{\Phi}_t \end{bmatrix} \boldsymbol{w}, \boldsymbol{\Sigma}_y\right) = \mathcal{N}\left(\boldsymbol{y}_t \middle| \boldsymbol{\Psi}_t \boldsymbol{w}, \boldsymbol{\Sigma}_y\right) \tag{10}$$

where $\boldsymbol{y}_{i,t} = [q_{i,t}, \dot{q}_{i,t}]^T$ denotes the joint angle and velocity for the i^{th} joint. We now maintain a distribution $p(\boldsymbol{w}; \boldsymbol{\theta})$ over the combined parameter vector $\boldsymbol{w}$. By introducing $p(\boldsymbol{w}; \boldsymbol{\theta})$, we extended our representation to additionally capture the correlation between the joints. The extended multi-DoF representation is used throughout the rest of the paper, including the experimental section. Controlling the robot in a coordinated manner using the coupling between the joints, for example, allows the robot to reach a via-point defined in the task-space while the joints exhibit variability. In the multi-DoF model, Eq. (1) becomes

$$p(\boldsymbol{\tau}|\boldsymbol{w}) = \prod_t \mathcal{N}\left(\boldsymbol{y}_t \middle| \boldsymbol{\Psi}_t \boldsymbol{w}, \boldsymbol{\Sigma}_y\right). \tag{11}$$

Additionally, our model captures the covariance of joint positions and velocities for each time step. Therefore, it encodes a linear relationship between them and enables to compute the desired velocity if the position is known or vice versa. We further exploit this property in Sect. 4.3.1 for adaptation to novel situations.

4.2 Learning from demonstrations

To simplify the learning of the parameters $\boldsymbol{\theta}$, we will assume a Gaussian distribution for $p(\boldsymbol{w}; \boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{w}|\boldsymbol{\mu}_w, \boldsymbol{\Sigma}_w)$ over the parameters $\boldsymbol{w}$. Consequently, the distribution of the state $p(\boldsymbol{y}_t|\boldsymbol{\theta})$ for time step t is given by

$$p\left(\boldsymbol{y}_t; \boldsymbol{\theta}\right) = \int \mathcal{N}\left(\boldsymbol{y}_t \middle| \boldsymbol{\Psi}_t \boldsymbol{w}, \boldsymbol{\Sigma}_y\right) \mathcal{N}\left(\boldsymbol{w} \middle| \boldsymbol{\mu}_{\boldsymbol{w}}, \boldsymbol{\Sigma}_{\boldsymbol{w}}\right) d\boldsymbol{w} = \mathcal{N}\left(\boldsymbol{y}_t \middle| \boldsymbol{\Psi}_t \boldsymbol{\mu}_{\boldsymbol{w}}, \boldsymbol{\Psi}_t \boldsymbol{\Sigma}_{\boldsymbol{w}} \boldsymbol{\Psi}_t^T + \boldsymbol{\Sigma}_y\right), \tag{12}$$

and, thus, we can easily evaluate the mean and the variance for any time point t. As a ProMP represents multiple ways to execute an elemental movement, we need multiple demonstrations in order to learn $p(\boldsymbol{w}; \boldsymbol{\theta})$, or, in the special case that only one demonstration is available, a prior variance profile for $p(\boldsymbol{w})$ should be given.[2]

4.2.1 Learning stroke-based movements

For stroke-based movements, we can estimate the parameters $\boldsymbol{\theta} = \{\boldsymbol{\mu}_{\boldsymbol{w}}, \boldsymbol{\Sigma}_{\boldsymbol{w}}\}$ from demonstrations by a simple maximum likelihood estimation algorithm. We estimate the weights for each trajectory individually with linear ridge regression, i.e.

$$\boldsymbol{w}_i = \left(\boldsymbol{\Psi}^T \boldsymbol{\Psi} + \lambda \boldsymbol{I}\right)^{-1} \boldsymbol{\Psi}^T \boldsymbol{Y}_i \tag{13}$$

where $\boldsymbol{Y}_i$ represents the positions of all joints and time steps from the demonstration i, and $\boldsymbol{\Psi}$ the corresponding basis function matrix for all time steps. We align the demonstrations by adjusting the phase signal. For each demonstration, we assume that $z_{\text{begin}} = 0$ and at the end $z_{\text{end}} = 1$. The ridge factor λ is generally set to a very small value, typically $\lambda = 10^{-12}$, as larger values degrade the estimation the trajectory distribution. In this paper, we also propose a new regularization scheme that is based on minimizing the jerk of the trajectories, i.e.,

$$\boldsymbol{w}_i = \left(\boldsymbol{\Psi}\boldsymbol{\Psi} + \lambda \boldsymbol{\Gamma}^T \boldsymbol{\Gamma}\right)^{-1} \boldsymbol{\Psi}^T \boldsymbol{Y}_i, \tag{14}$$

where $\boldsymbol{\Gamma}$ denotes the third derivative[3] of $\boldsymbol{\Psi}$. The third derivative is needed as the jerk is given by the third derivative. The jerk minimization scheme can generate smoother torque profiles and, hence, performs better in the cost function comparison presented in Table 2. The mean $\boldsymbol{\mu}_{\boldsymbol{w}}$ and covariance $\boldsymbol{\Sigma}_{\boldsymbol{w}}$ are computed from the samples $\boldsymbol{w}_i$,

$$\boldsymbol{\mu}_{\boldsymbol{w}} = \frac{1}{N}\sum_{i=1}^{N} \boldsymbol{w}_i, \quad \hat{\boldsymbol{\Sigma}}_{\boldsymbol{w}} = \frac{1}{N}\sum_{i=1}^{N} (\boldsymbol{w}_i - \boldsymbol{\mu}_{\boldsymbol{w}})(\boldsymbol{w}_i - \boldsymbol{\mu}_{\boldsymbol{w}})^T \tag{15}$$

where N is the number of demonstrations. We use an Inverse-Wishart distribution as a prior to the covariance matrix $\boldsymbol{\Sigma}_{\boldsymbol{w}}$.

[2] This prior variance profile can be just set to $\alpha \boldsymbol{I}$, where α is a small constant and $\boldsymbol{I}$ is the identity matrix.

[3] The third derivative of $\boldsymbol{\Psi}$ can be computed numerically.

The maximum a-posteriori estimate of the covariance (OHagan and Forster 2004) given the prior becomes

$$\boldsymbol{\Sigma}_{\boldsymbol{w}} = \frac{N\hat{\boldsymbol{\Sigma}}_{\boldsymbol{w}} + \lambda_{\boldsymbol{w}}\boldsymbol{I}}{N+\lambda}, \tag{16}$$

where the value of $\lambda_{\boldsymbol{w}}$ is set such that the covariance matrix $\boldsymbol{\Sigma}_{\boldsymbol{w}}$ is positive-definite. The complete algorithm is shown in Algorithm 1.

4.2.2 Learning periodic movements

In this section we present an Expectation-Maximization (EM) algorithm that can be used to learn from missing data or rhythmic movements. Using the previous learning approach for periodic movements would require that each demonstration finishes at the same state as it started, as we use a single weight vector per demonstration and the basis functions are periodic. However, due to the variability, single trajectories typically do not end exactly where they started. Yet, rhythmic movements can be learned by using an EM-algorithm that we can train with partial trajectories, i.e., trajectories that do not cover a whole period.

We derive an Expectation Maximization (EM) algorithm that infers the latent variables, i.e. the weights for each demonstrations during training (Ewerton et al. 2015). We assume that our set of demonstrations contains multiple periods. First, we determine the period length from the demonstration and we construct the basis and phase signal. We randomly split the demonstration to N potentially overlapping segments. The size of the segment must be shorter than a period to avoid the periodicity in the basis functions for a single demonstration. The initial guess for the parameters is estimated using linear ridge regression. In the expectation step, we need to compute the posterior distribution of the weights

$$p(\boldsymbol{w}_i|\boldsymbol{Y}_i, \boldsymbol{\mu}_{\boldsymbol{w}}, \boldsymbol{\Sigma}_{\boldsymbol{w}}) \propto p(\boldsymbol{Y}_i|\boldsymbol{w}_i)p(\boldsymbol{w}_i|\boldsymbol{\mu}_{\boldsymbol{w}}, \boldsymbol{\Sigma}_{\boldsymbol{w}}), \tag{17}$$

for each demonstration. The posterior can be computed using the Bayes rule for Gaussian distributions. The expectation step becomes

$$\boldsymbol{\mu}_i = \boldsymbol{\mu}_{\boldsymbol{w}} + \boldsymbol{\Psi}_i^T \left(\boldsymbol{\Psi}_i \boldsymbol{\Sigma}_{\boldsymbol{w}} \boldsymbol{\Psi}_i^T\right)^{-1} (\boldsymbol{Y}_i - \boldsymbol{\Psi}_i \boldsymbol{\mu}_w), \tag{18}$$

$$\boldsymbol{\Sigma}_i = \boldsymbol{\Sigma}_{\boldsymbol{w}} - \boldsymbol{\Sigma}_{\boldsymbol{w}} \boldsymbol{\Psi}_i^T \left(\boldsymbol{\Psi}_i \boldsymbol{\Sigma}_{\boldsymbol{w}} \boldsymbol{\Psi}_i^T\right)^{-1} \boldsymbol{\Psi}_i \boldsymbol{\Sigma}_{\boldsymbol{w}}, \tag{19}$$

where the index i denotes the i-th segment of the demonstration and $\boldsymbol{\Psi}_i$ the basis functions for that segment. We dropped the time dependency from the notation of $\boldsymbol{\Psi}_i$ for clearness. In the maximization step, we need to optimize the complete-data log-likelihood

$$\text{argmax}_{\boldsymbol{\theta}'} \sum_{i=1}^{N} \int_w p\left(\boldsymbol{w}_i|\boldsymbol{\theta}\right) \log p\left(\boldsymbol{Y}_i\middle|\boldsymbol{\theta}'\right) p\left(\boldsymbol{w}\middle|\boldsymbol{\theta}'\right) d\boldsymbol{w} \tag{20}$$

where $\boldsymbol{\theta}' = \{\boldsymbol{\mu}'_{\boldsymbol{w}}, \boldsymbol{\Sigma}'_{\boldsymbol{w}}\}$ denote the new parameters for the weight distribution. Thus, the maximization step becomes

$$\boldsymbol{\mu}'_{\boldsymbol{w}} = \frac{1}{N}\sum_{i=1}^{N} \boldsymbol{\mu}_i, \tag{21}$$

$$\boldsymbol{\Sigma}'_{\boldsymbol{w}} = \frac{1}{N}\sum_{i=1}^{N} \left(\left(\boldsymbol{\mu}_i - \boldsymbol{\mu}'_{\boldsymbol{w}}\right)\left(\boldsymbol{\mu}_i - \boldsymbol{\mu}'_{\boldsymbol{w}}\right)^T + \boldsymbol{\Sigma}_i\right), \tag{22}$$

for computing the updates in closed form. We iterate between the expectation step and the maximization step until convergence. Our algorithm is based on the EM from HBMs with Gaussian distributions approach presented in Lazaric and Ghavamzadeh (2010) and has been evaluated in Paraschos et al. (2013a) and Ewerton et al. (2015) for the ProMP representation. The algorithm for learning periodic movements is shown in Algorithm 2.

In both learning approaches, the weight covariance $\boldsymbol{\Sigma}_{\boldsymbol{w}}$ may become not positive definite because of numerical problems. To correct these numerical problems we use an eigen-decomposition to find the closest symmetric positive definite matrix to our estimation, as described in Higham (1988).

4.3 New probabilistic operators for movement primitives

With the probabilistic representation we can exploit probabilistic operators, i.e., modulate the trajectory by conditioning and co-activate MPs by computing the product of distributions.

Using Gaussian distributions for $p(\boldsymbol{w}; \boldsymbol{\theta})$, all operators can be computed in closed form.

4.3.1 Modulation of the trajectory distribution by conditioning

The modulation of via-points and final positions is an important property to adapt the MP to new situations. In our probabilistic formulation, such operations can be described by conditioning the MP to reach a certain state $\boldsymbol{y}_t^*$ at time t. Note that conditioning can be performed for any time point t. It is performed by adding a desired observation

$$\boldsymbol{x}_t^* = \left\{\boldsymbol{y}_t^*, \boldsymbol{\Sigma}_y^*\right\} \tag{23}$$

to our probabilistic model and applying Bayes theorem, i.e.

$$p\left(\boldsymbol{w}\middle|\boldsymbol{x}_t^*\right) \propto \mathcal{N}\left(\boldsymbol{y}_t^*\middle|\boldsymbol{\Psi}_t\boldsymbol{w}, \boldsymbol{\Sigma}_y^*\right) p(\boldsymbol{w}), \tag{24}$$

Algorithm 2: Learning Periodic Movements

Data: A trajectory with multiple periods with position observations $\boldsymbol{Y}$, at time t
Input: Number of basis functions K, Basis function width b, Regression parameter λ, Number of segments to split N, EM convergence parameter ϵ
Result: The mean $\boldsymbol{\mu}_{\boldsymbol{w}}$ and covariance $\boldsymbol{\Sigma}_{\boldsymbol{w}}$ of $p(\boldsymbol{w}) \sim \mathcal{N}\left(\boldsymbol{w}\middle|\boldsymbol{\mu}_{\boldsymbol{w}}, \boldsymbol{\Sigma}_{\boldsymbol{w}}\right)$
→ Detect base frequency: f_q by FFT;
→ Periodic phase signal: $z = \mathbf{mod}(t f_q, 1)$;
→ Split randomly: $\{\boldsymbol{Y}, z\}$ into N segments;
→ Initial guess: $\boldsymbol{\mu}_{\boldsymbol{w}}$ and $\boldsymbol{\Sigma}_{\boldsymbol{w}}$ from Algorithm 1;
repeat
Expectation step:

$$\boldsymbol{\mu}_i = \boldsymbol{\mu}_{\boldsymbol{w}} + \boldsymbol{\Psi}_i^T \left(\boldsymbol{\Psi}_i \boldsymbol{\Sigma}_{\boldsymbol{w}} \boldsymbol{\Psi}_i^T\right)^{-1} \left(\boldsymbol{Y}_i - \boldsymbol{\Psi}_i \boldsymbol{\mu}_w\right),$$

$$\boldsymbol{\Sigma}_i = \boldsymbol{\Sigma}_{\boldsymbol{w}} - \boldsymbol{\Sigma}_{\boldsymbol{w}} \boldsymbol{\Psi}_i^T \left(\boldsymbol{\Psi}_i \boldsymbol{\Sigma}_{\boldsymbol{w}} \boldsymbol{\Psi}_i^T\right)^{-1} \boldsymbol{\Psi}_i \boldsymbol{\Sigma}_{\boldsymbol{w}}$$

Maximization step:

$$\boldsymbol{\mu}'_{\boldsymbol{w}} = \frac{1}{N} \sum_{i=1}^{N} \boldsymbol{\mu}_i,$$

$$\boldsymbol{\Sigma}'_{\boldsymbol{w}} = \frac{1}{N} \sum_{i=1}^{N} \left(\left(\boldsymbol{\mu}_i - \boldsymbol{\mu}'_{\boldsymbol{w}}\right)\left(\boldsymbol{\mu}_i - \boldsymbol{\mu}'_{\boldsymbol{w}}\right)^T + \boldsymbol{\Sigma}_i\right)$$

until *difference in log-likelihood* $< \epsilon$;
return $\boldsymbol{\mu}'_{\boldsymbol{w}}$, $\boldsymbol{\Sigma}'_{\boldsymbol{w}}$.

where the state vector $\boldsymbol{y}_t^*$ represents the desired position and velocity vector at time t and $\boldsymbol{\Sigma}_y^*$ describes the accuracy of the desired observation. We can also condition on any subset of $\boldsymbol{y}_t^*$. For example, specifying a desired joint position q_1 for the first joint the trajectory distribution will automatically infer the most probable joint positions for the other joints. Conditioning partially on the state is done by constructing the basis function matrix $\boldsymbol{\Psi}$ used in Eqs. (25) and (26) to contain only the variables that participate in the conditioning. For example, Maeda et al. (2014) used such an approach based on ProMPs to model human–robot interaction where conditioning on the human movement yields the desired movement of the robot.

For Gaussian trajectory distributions, the conditional distribution $p\left(\boldsymbol{w}\middle|\boldsymbol{x}_t^*\right)$ for $\boldsymbol{w}$ is Gaussian with mean and variance

$$\boldsymbol{\mu}_{\boldsymbol{w}}^{[\text{new}]} = \boldsymbol{\mu}_{\boldsymbol{w}} + \boldsymbol{L}\left(\boldsymbol{y}_t^* - \boldsymbol{\Psi}_t^T \boldsymbol{\mu}_{\boldsymbol{w}}\right), \tag{25}$$

$$\boldsymbol{\Sigma}_{\boldsymbol{w}}^{[\text{new}]} = \boldsymbol{\Sigma}_{\boldsymbol{w}} - \boldsymbol{L}\boldsymbol{\Psi}_t^T \boldsymbol{\Sigma}_{\boldsymbol{w}}, \tag{26}$$

where $\boldsymbol{L}$ is given by

$$\boldsymbol{L} = \boldsymbol{\Sigma}_{\boldsymbol{w}} \boldsymbol{\Psi}_t \left(\boldsymbol{\Sigma}_y^* + \boldsymbol{\Psi}_t^T \boldsymbol{\Sigma}_{\boldsymbol{w}} \boldsymbol{\Psi}_t\right)^{-1}. \tag{27}$$

Illustrative Example Conditioning a ProMP to different target states, positions and velocities, is illustrated in Fig. 6. We observe that, despite the modulation of the ProMP by conditioning, the ProMP stays within the original distribution. How the ProMPs modulate is hence learned from the original demonstrations. Modulation strategies in other approaches such as the DMPs do not show this effect (Schaal et al. 2005). DMPs can reach the desired target position and velocities at the end of the movement, but deform the trajectory significantly. In contrast, the trajectory distribution obtained by conditioning a ProMP even matches the distribution of the optimal controller that has the conditioned via-point as additional cost term.

4.3.2 Adaptation to task parameters

In many situations, we need to adapt the primitive based on an external state variable $\hat{\boldsymbol{s}}$, such as a desired target angle when shooting hockey pucks. The value of such external variables is typically known during training and also before reproduction of the primitive. Hence, we can directly learn this adaptation by learning a mapping from the external variable to the mean weight vector $\boldsymbol{\mu}_{\boldsymbol{w}}$. We use a simple linear mapping, which is equivalent to modeling a joint distribution

$$\begin{aligned} p\left(\boldsymbol{w}, \hat{\boldsymbol{s}}\right) &= \mathcal{N}\left(\begin{bmatrix}\boldsymbol{w} \\ \hat{\boldsymbol{s}}\end{bmatrix}\middle|\boldsymbol{\mu}, \boldsymbol{\Sigma}\right) \\ &= \mathcal{N}\left(\boldsymbol{w}\middle|\boldsymbol{O}\hat{\boldsymbol{s}} + \boldsymbol{o}, \boldsymbol{\Sigma}_{\boldsymbol{w}}\right) \mathcal{N}\left(\hat{\boldsymbol{s}}\middle|\boldsymbol{\mu}_{\hat{\boldsymbol{s}}}, \boldsymbol{\Sigma}_{\hat{\boldsymbol{s}}}\right), \end{aligned} \tag{28}$$

however, the transformation parameters $\{\boldsymbol{O}, \boldsymbol{o}\}$ are learned directly with linear ridge regression.

4.3.3 Combination and blending of movement primitives

We can use a product of trajectory distributions to continuously combine and blend different MPs into a single movement. Suppose that we maintain a set of i different primitives that we want to combine. We can co-activate them by taking the products of distributions,

$$p_{\text{new}}(\boldsymbol{\tau}) \propto \textstyle\prod_i p_i(\boldsymbol{\tau})^{\alpha^{[i]}}, \tag{29}$$

where the $\alpha^{[i]} \in [0, 1]$ factors denote the activation of the ith primitive. The product captures the overlapping region of the active MPs, i.e., the part of the trajectory space where all MPs have high probability mass.

We also want to be able to modulate the activations of the primitives, for example, to continuously blend the movement execution from one primitive to the next one. Hence, we decompose the trajectory into its single time steps and use time-varying activation functions $\alpha_t^{[i]}$, i.e.,

$$p^*(\boldsymbol{\tau}) \propto \prod_t \prod_i p_i(\boldsymbol{y}_t)^{\alpha_t^{[i]}}, \tag{30}$$

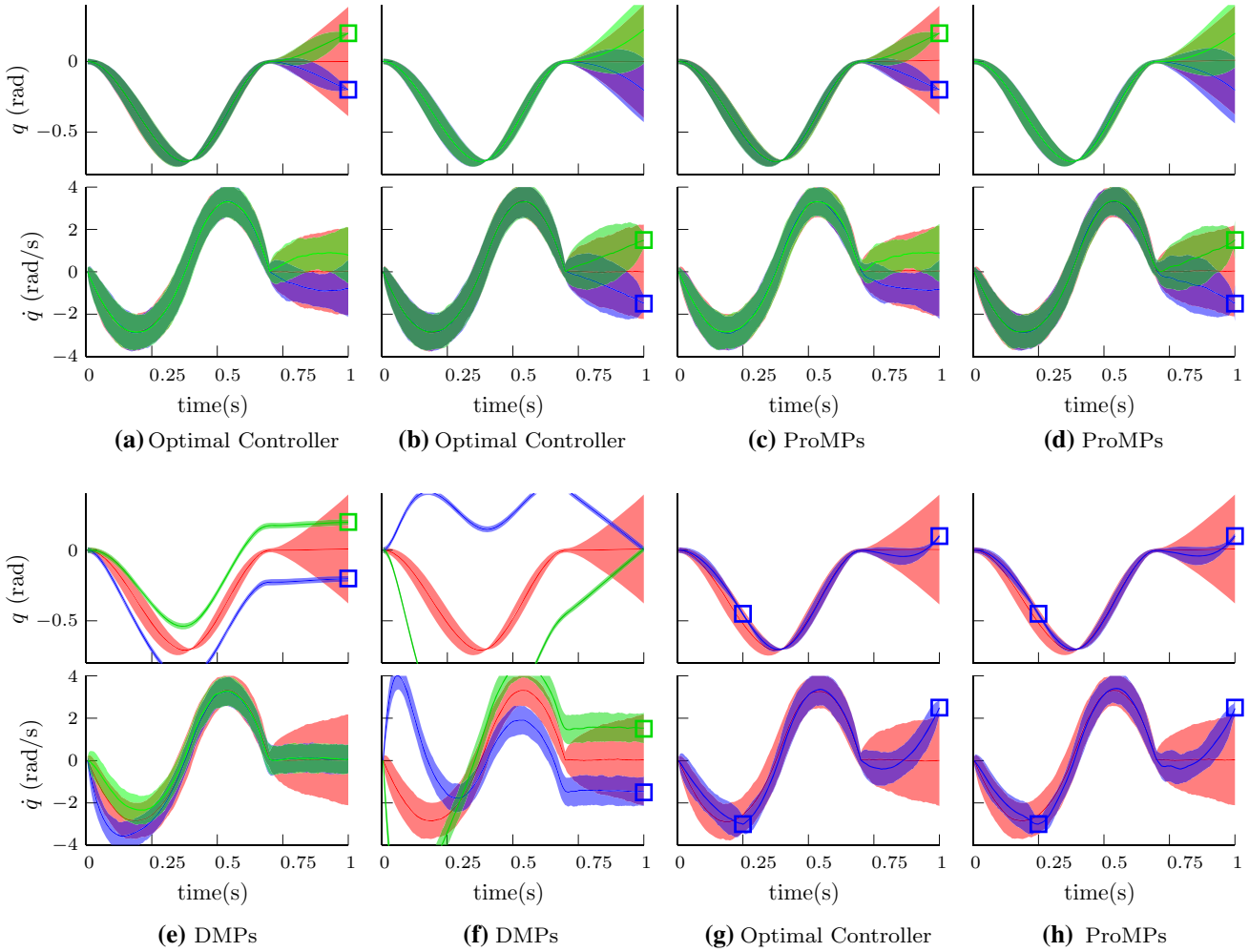


Fig. 6 *Generalization of primitives*. We want to modulate the MPs such that they go through additional via-points (*blue* and *green*) and evaluate the quality of the generalized MP policies. The resulting distributions are illustrated only for comparison and are not used for training. The added via-points are depicted with *colored boxes*. **a**, **b** Evaluation of the optimal controller given the additional via-points on the final position (**a**) or final velocity (**b**). **c**, **d** Evaluation of the ProMP on the same via-points. ProMPs reproduce the optimal behavior despite that the unconditioned demonstrations have been used for training. **e**, **f** Generalization to the same via-points with DMPs. The position generalization is a linear interpolation of the mean trajectory and quickly goes "outside" the demonstrated distribution. The final velocity generalization reproduce drastically different trajectories than the demonstrated ones. **g**, **h** Evaluation of the optimal controller and the ProMPs on additional via-point in intermediate and final locations, that require adaptation on both the position and the velocity simultaneously (Color figure online)

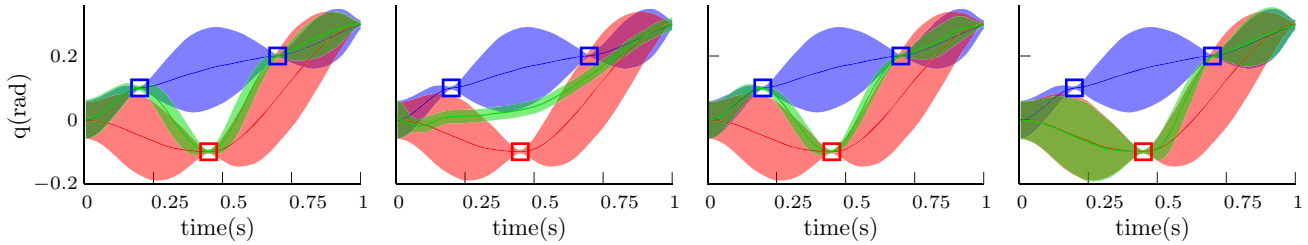
$$p_i(\boldsymbol{y}_t) = \int p_i\left(\boldsymbol{y}_t \middle| \boldsymbol{w}^{[i]}\right) p_i\left(\boldsymbol{w}^{[i]}\right) d\boldsymbol{w}^{[i]}. \tag{31}$$

For Gaussian distributions $p_i(\boldsymbol{y}_t) = \mathcal{N}(\boldsymbol{y}_t|\boldsymbol{\mu}_t^{[i]}, \boldsymbol{\Sigma}_t^{[i]})$, the resulting distribution $p^*(\boldsymbol{y}_t)$ is again Gaussian with variance and mean,

$$\boldsymbol{\Sigma}_t^* = \left(\sum_i \left(\boldsymbol{\Sigma}_t^{[i]}/\alpha_t^{[i]}\right)^{-1}\right)^{-1}, \tag{32}$$

$$\boldsymbol{\mu}_t^* = \boldsymbol{\Sigma}_t^* \left(\sum_i \left(\boldsymbol{\Sigma}_t^{[i]}/\alpha_t^{[i]}\right)^{-1} \boldsymbol{\mu}_t^{[i]}\right). \tag{33}$$

Illustrative Example. Co-activation of two ProMPs is shown in Fig. 7c and blending of two ProMPs in Fig. 7d. We trained the ProMPs such that each primitive solves a different task indicated by the via points in the figures with the same colors. The combined primitive is capable of reaching all four via-points, i.e., it achieved both tasks at the *same* time. Additionally, we compare our combination approach to the optimal controller by adding the cost functions of the two tasks. The optimal controller results are shown in Fig. 7a. Combining movements with the DMPs results on averaging between the trajectories and therefore missing all of the via-points. The trajectory distribution is shown in Fig. 7b. We quantified the results in terms of the average cost in Table 2.



(a) Opt. Ctl. Combination **(b)** DMP Combination **(c)** ProMP Combination **(d)** ProMP Blending

Fig. 7 *Combination and blending of two primitives*. We want to combine two MPs to obtain an MP that can achieve both tasks of the single MPs at the same time. We show the resulting distribution in *green* and the participating primitives in *blue* and *red*. **a** The resulting optimal distribution is generated by adding both cost-functions that have been used to generate the single primitive distributions. **b** Combining DMPs linearly in weight space results in a linearly interpolated trajectory. The movement misses all the via-points. **c** We co-activate two ProMPs with equal weights. The resulting movement passes through all via-points. **d** We smoothly blend from the *red* primitive to the *blue* primitive. The resulting movement (*green*) first follows the *red* primitive and, subsequently, switches to following exactly the *blue* primitive (Color figure online)

While the ProMP approach achieves an average cost in the same range of magnitude, the performance of the DMP combination is highly degraded.

4.4 Using trajectory distributions for robot control

In order to fully exploit the properties of trajectory distributions, a policy that reproduces these distributions is needed for controlling the robot. To this effect, we derive a stochastic feedback controller that can accurately reproduce the mean $\boldsymbol{\mu}_t$, the variances $\boldsymbol{\Sigma}_t$, and the correlations $\boldsymbol{\Sigma}_{t,t+1}$ for all time steps t of a given trajectory distribution. The derivation of the controller is based on moment matching on Gaussian distribution. In our approach there is no notion of cost function.

Such controller can only be obtained by using a model. We approximate the continuous time dynamics of the system by a linearized discrete-time system with step duration dt,

$$\boldsymbol{y}_{t+\mathrm{dt}} = (\boldsymbol{I} + \boldsymbol{A}_t\,\mathrm{dt})\,\boldsymbol{y}_t + \boldsymbol{B}_t\,\mathrm{dt}\,\boldsymbol{u} + \boldsymbol{c}_t\,\mathrm{dt}, \tag{34}$$

where the system matrices $\boldsymbol{A}_t$, the input matrices $\boldsymbol{B}_t$ and the drift vectors $\boldsymbol{c}_t$ can be obtained by first order Taylor expansion of the dynamical system for the current state $\boldsymbol{y}_t$.[4] We assume a stochastic linear feedback controller with time varying feedback gains is generating the control actions, i.e.,

$$\boldsymbol{u} = \boldsymbol{K}_t\,\boldsymbol{y}_t + \boldsymbol{k}_t + \boldsymbol{\epsilon}_{\boldsymbol{u}}, \quad \boldsymbol{\epsilon} \sim \mathcal{N}\left(\boldsymbol{\epsilon}_{\boldsymbol{u}}\middle|0, \boldsymbol{\Sigma}_{\boldsymbol{u}}\mathrm{dt}^{-1}\right), \tag{35}$$

where the matrix $\boldsymbol{K}_t$ denotes a feedback gain matrix and $\boldsymbol{k}_t$ a feed-forward component. We use a control noise which behaves like a Wiener process (Stark and Woods 2001), and, hence, its variance grows linearly with the step duration[5] dt. By substituting Eq. (35) into Eq. (34), we can rewrite the next state of the system as

$$\begin{aligned}\boldsymbol{y}_{t+\mathrm{dt}} &= (\boldsymbol{I} + (\boldsymbol{A}_t + \boldsymbol{B}_t\boldsymbol{K}_t)\,\mathrm{dt})\,\boldsymbol{y}_t \\ &\quad + \boldsymbol{B}_t\,\mathrm{dt}(\boldsymbol{k}_t + \boldsymbol{\epsilon}_u) + \boldsymbol{c}\,\mathrm{dt} \\ &= \boldsymbol{F}_t\boldsymbol{y}_t + \boldsymbol{f}_t + \boldsymbol{B}_t\,\mathrm{dt}\,\boldsymbol{\epsilon}_u,\end{aligned} \tag{36}$$

where we defined

$$\begin{aligned}\boldsymbol{F}_t &= (\boldsymbol{I} + (\boldsymbol{A}_t + \boldsymbol{B}_t\boldsymbol{K}_t)\,\mathrm{dt})\,, \\ \boldsymbol{f}_t &= \boldsymbol{B}_t\boldsymbol{k}_t\,\mathrm{dt} + \boldsymbol{c}\,\mathrm{dt}\,.\end{aligned} \tag{37}$$

We will omit the time-index as subscript for most matrices in the remainder of the paper to improve readability. From Eq. (12), we know that the distribution for our current state $\boldsymbol{y}_t$ is Gaussian with mean $\boldsymbol{\mu}_t = \boldsymbol{\Psi}_t\boldsymbol{\mu}_w$ and covariance[6] $\boldsymbol{\Sigma}_t = \boldsymbol{\Psi}_t\boldsymbol{\Sigma}_{\boldsymbol{w}}\boldsymbol{\Psi}_t^T$. As the system dynamics are modeled by a Gaussian linear model, we can obtain the distribution of the next state $p\,(\boldsymbol{y}_{t+\mathrm{dt}})$ analytically from the forward model by integrating out the current state

$$\begin{aligned}p(\boldsymbol{y}_{t+\mathrm{dt}}) &= \int_{\boldsymbol{y}_t} \mathcal{N}\left(\boldsymbol{y}_{t+\mathrm{dt}}\middle|\boldsymbol{F}\boldsymbol{y}_t + \boldsymbol{f}, \boldsymbol{\Sigma}_s\,\mathrm{dt}\right)\mathcal{N}\left(\boldsymbol{y}_t\middle|\boldsymbol{\mu}_t, \boldsymbol{\Sigma}_t\right) \\ &= \mathcal{N}\left(\boldsymbol{y}_{t+\mathrm{dt}}\middle|\boldsymbol{F}\boldsymbol{\mu}_t + \boldsymbol{f}, \boldsymbol{F}\boldsymbol{\Sigma}_t\boldsymbol{F}^T + \boldsymbol{\Sigma}_s\,\mathrm{dt}\right),\end{aligned} \tag{38}$$

where $\mathrm{dt}\,\boldsymbol{\Sigma}_s = \mathrm{dt}\,\boldsymbol{B}\boldsymbol{\Sigma}_u\boldsymbol{B}^T$ represents the system noise matrix. Both sides of Eq. (38) are Gaussian distributions. The left-hand side can be computed in two ways; from our desired trajectory distribution $p(\boldsymbol{\tau};\boldsymbol{\theta})$ and from Eq. (38). We

[4] If inverse dynamics control (Peters et al. 2008) is used for the robot, the system reduces to a linear system where the terms $\boldsymbol{A}_t$, $\boldsymbol{B}_t$ and $\boldsymbol{c}_t$ are constant in time.

[5] As we multiply the noise by $\boldsymbol{B}\,\mathrm{dt}$, we need to divide the covariance $\boldsymbol{\Sigma}_u$ of the control noise $\boldsymbol{\epsilon}_u$ by dt to obtain this desired behavior.

[6] The observation noise is omitted as it represents independent noise which is not used for predicting the next state.

proceed by matching the mean and the variances of both sides with our control law,

$$\boldsymbol{\mu}_{t+\mathrm{dt}} = \boldsymbol{F}\boldsymbol{\mu}_t + (\boldsymbol{B}\boldsymbol{k} + \boldsymbol{c})\,\mathrm{dt}, \tag{39}$$

$$\boldsymbol{\Sigma}_{t+\mathrm{dt}} = \boldsymbol{F}\boldsymbol{\Sigma}_t\boldsymbol{F}^T + \boldsymbol{\Sigma}_s\,\mathrm{dt}, \tag{40}$$

where $\boldsymbol{F}$ is given in Eq. (37) and contains the time varying feedback gains $\boldsymbol{K}$. Using both constraints, we can now obtain the time-dependent gains $\boldsymbol{K}_t$ and $\boldsymbol{k}_t$. Note that the linearized model given by $\boldsymbol{A}_t$, $\boldsymbol{B}_t$ and $\boldsymbol{c}_t$ depends on the current state $\boldsymbol{y}_t$ which is used as linearization point. As our computation of the gains will depend on the linearized model, our controller gains also depend implicitely on the current state, i.e., $\boldsymbol{K}_t = \boldsymbol{K}(\boldsymbol{y}_t)$ and $\boldsymbol{k}_t = \boldsymbol{k}(\boldsymbol{y}_t)$. Therefore, our controller is in fact a non-linear controller. However, we will ommit the state dependence of our gains in the remaining derivation for the sake of clarity.

4.4.1 Derivation of the controller gains

We continue with the derivation of the controller gains, $\boldsymbol{K}$. To perform the derivation we assume, for the moment, that the stochasticity of the controller $\boldsymbol{\Sigma}_u$ is known. In Sect. 4.4.3, we show how the stochasticity of the controller can be computed closed form. By rearranging terms, the covariance constraint becomes

$$\begin{aligned}\boldsymbol{\Sigma}_{t+\mathrm{dt}} - \boldsymbol{\Sigma}_t = \boldsymbol{\Sigma}_s\,\mathrm{dt} + (\boldsymbol{A} + \boldsymbol{B}\boldsymbol{K})\,\boldsymbol{\Sigma}_t\,\mathrm{dt} \\ + \boldsymbol{\Sigma}_t\,(\boldsymbol{A} + \boldsymbol{B}\boldsymbol{K})^T\,\mathrm{dt} + O(\mathrm{dt}^2),\end{aligned} \tag{41}$$

where $O(\mathrm{dt}^2)$ denotes all second order terms in dt. After dividing by dt and taking the limit of $\mathrm{dt} \to 0$, the second order terms disappear and we obtain the time derivative of the covariance

$$\begin{aligned}\dot{\boldsymbol{\Sigma}}_t &= \lim_{\mathrm{dt}\to 0} \frac{\boldsymbol{\Sigma}_{t+\mathrm{dt}} - \boldsymbol{\Sigma}_t}{\mathrm{dt}} \\ &= (\boldsymbol{A} + \boldsymbol{B}\boldsymbol{K})\,\boldsymbol{\Sigma}_t + \boldsymbol{\Sigma}_t\,(\boldsymbol{A} + \boldsymbol{B}\boldsymbol{K})^T + \boldsymbol{\Sigma}_s,\end{aligned} \tag{42}$$

which is a special case of the continuous time Ricatti equation. Note that this operation was only possible due to the continuous time formulation of the basis functions.

The derivative of the covariance matrix $\dot{\boldsymbol{\Sigma}}_t$ can additionally be obtained from the trajectory distribution by

$$\dot{\boldsymbol{\Sigma}}_t = \dot{\boldsymbol{\Psi}}_t\boldsymbol{\Sigma}_w\boldsymbol{\Psi}_t^T + \boldsymbol{\Psi}_t\boldsymbol{\Sigma}_w\dot{\boldsymbol{\Psi}}_t^T, \tag{43}$$

which we substitute into Eq. (42). After rearranging terms, the equation reads

$$\boldsymbol{M} + \boldsymbol{M}^T = \boldsymbol{B}\boldsymbol{K}\boldsymbol{\Sigma}_t + (\boldsymbol{B}\boldsymbol{K}\boldsymbol{\Sigma}_t)^T, \tag{44}$$

where we defined

$$\boldsymbol{M} = \dot{\boldsymbol{\Psi}}_t\boldsymbol{\Sigma}_w\boldsymbol{\Psi}_t^T - \boldsymbol{A}\boldsymbol{\Sigma}_t - 0.5\boldsymbol{\Sigma}_s, \tag{45}$$

to demonstrate the structure of the equation. A solution can be obtained by setting $\boldsymbol{M} = \boldsymbol{B}\boldsymbol{K}\boldsymbol{\Sigma}_t$ and solving for the gain matrix $\boldsymbol{K}$,

$$\boldsymbol{K} = \boldsymbol{B}^{\dagger}\left(\dot{\boldsymbol{\Psi}}_t\boldsymbol{\Sigma}_w\boldsymbol{\Psi}_t^T - \boldsymbol{A}\boldsymbol{\Sigma}_t - 0.5\boldsymbol{\Sigma}_s\right)\boldsymbol{\Sigma}_t^{-1}, \tag{46}$$

where $\boldsymbol{B}^{\dagger}$ denotes the pseudo-inverse of the control matrix $\boldsymbol{B}$.

4.4.2 Derivation of the feed-forward controls

Similarly, we obtain the feed-forward control signal $\boldsymbol{k}$ by matching the mean of the trajectory distribution $\boldsymbol{\mu}_{t+\mathrm{dt}}$ with the mean computed with the forward model. After rearranging terms, dividing by dt, and taking the limit of $\mathrm{dt} \to 0$, we arrive at

$$\dot{\boldsymbol{\mu}}_t = (\boldsymbol{A} + \boldsymbol{B}\boldsymbol{K})\,\boldsymbol{\mu}_t + \boldsymbol{B}\boldsymbol{k} + \boldsymbol{c}, \tag{47}$$

the differential equation for the mean of the trajectory. We use the trajectory distribution $p(\boldsymbol{\tau};\boldsymbol{\theta})$ to obtain $\boldsymbol{\mu}_t = \boldsymbol{\Psi}_t\boldsymbol{\mu}_w$ and $\dot{\boldsymbol{\mu}}_t = \dot{\boldsymbol{\Psi}}_t\boldsymbol{\mu}_w$ and solve Eq. (47) for $\boldsymbol{k}$,

$$\boldsymbol{k} = \boldsymbol{B}^{\dagger}\left(\dot{\boldsymbol{\Psi}}_t\boldsymbol{\mu}_w - (\boldsymbol{A} + \boldsymbol{B}\boldsymbol{K})\,\boldsymbol{\Psi}_t\boldsymbol{\mu}_w - \boldsymbol{c}\right). \tag{48}$$

The time-varying feedback gains $\boldsymbol{K}$ do not depend on the mean of the trajectory distribution, but only on the variance at that time step. Similarly, the feed-forward controls $\boldsymbol{k}$, depend on the variance only through the feedback gains $\boldsymbol{K}$, but otherwise they depend on the mean.

4.4.3 Estimation of the control noise

The last step required to match the trajectory distribution is to match the control noise matrix $\boldsymbol{\Sigma}_u$ which is needed to generate the distribution. This noise can be higher than the system noise to induce a higher variance in the distribution. Such a higher variance can, for example, be useful for exploration in reinforcement learning.

We compute the system noise covariance $\boldsymbol{\Sigma}_s = \boldsymbol{B}\boldsymbol{\Sigma}_u\boldsymbol{B}^T$ by examining the cross-correlation between time steps of the trajectory distribution. To do so, we compute the joint distribution $p\left(\boldsymbol{y}_t, \boldsymbol{y}_{t+\mathrm{dt}}\right)$ of the current state $\boldsymbol{y}_t$ and the next state $\boldsymbol{y}_{t+\mathrm{dt}}$ as

$$\begin{aligned}&p\left(\boldsymbol{y}_t, \boldsymbol{y}_{t+\mathrm{dt}}\right) \\ &= \mathcal{N}\left(\begin{bmatrix}\boldsymbol{y}_t \\ \boldsymbol{y}_{t+\mathrm{dt}}\end{bmatrix} \middle| \begin{bmatrix}\boldsymbol{\mu}_t \\ \boldsymbol{\mu}_{t+\mathrm{dt}}\end{bmatrix}, \begin{bmatrix}\boldsymbol{\Sigma}_t & \boldsymbol{C}_t \\ \boldsymbol{C}_t^T & \boldsymbol{\Sigma}_{t+\mathrm{dt}}\end{bmatrix}\right),\end{aligned} \tag{49}$$

where $\boldsymbol{C}_t = \boldsymbol{\Psi}_t \boldsymbol{\Sigma}_{\boldsymbol{w}} \boldsymbol{\Psi}_{t+\mathrm{dt}}^T$ is the cross-correlation of the subsequent time points. We use our linear Gaussian model to match the cross correlation. The joint distribution for $\boldsymbol{y}_t$ and $\boldsymbol{y}_{t+\mathrm{dt}}$ can also be obtained by our system dynamics, i.e.,

$$p\left(\boldsymbol{y}_t, \boldsymbol{y}_{t+\mathrm{dt}}\right) = \mathcal{N}\left(\boldsymbol{y}_t | \boldsymbol{\mu}_t, \boldsymbol{\Sigma}_t\right) \mathcal{N}\left(\boldsymbol{y}_{t+\mathrm{dt}} | \boldsymbol{F}\boldsymbol{y}_t + \boldsymbol{f}, \boldsymbol{\Sigma}_{\boldsymbol{u}}\right)$$

which yields a Gaussian distribution with mean and covariance

$$\hat{\boldsymbol{\mu}}_{\boldsymbol{t}} = \begin{bmatrix} \boldsymbol{\mu}_t \\ \boldsymbol{F}\boldsymbol{\mu}_t + \boldsymbol{f} \end{bmatrix}, \quad \hat{\boldsymbol{\Sigma}}_{\boldsymbol{t}} = \begin{bmatrix} \boldsymbol{\Sigma}_t & \boldsymbol{\Sigma}_t \boldsymbol{F}^T \\ \boldsymbol{F}\boldsymbol{\Sigma}_t & \boldsymbol{F}\boldsymbol{\Sigma}_t \boldsymbol{F}^T + \boldsymbol{\Sigma}_s \,\mathrm{dt}\,. \end{bmatrix} \tag{50}$$

The noise covariance $\boldsymbol{\Sigma}_s$ is obtained by matching both covariance matrices given in Eqs. (49) and (50),

$$\begin{aligned} \boldsymbol{\Sigma}_s \,\mathrm{dt} &= \boldsymbol{\Sigma}_{t+\mathrm{dt}} - \boldsymbol{F}\boldsymbol{\Sigma}_t \boldsymbol{F}^T = \boldsymbol{\Sigma}_{t+\mathrm{dt}} - \boldsymbol{F}\boldsymbol{\Sigma}_t \boldsymbol{\Sigma}_t^{-1} \boldsymbol{\Sigma}_t \boldsymbol{F}^T \\ &= \boldsymbol{\Sigma}_{t+\mathrm{dt}} - \boldsymbol{C}_t^T \boldsymbol{\Sigma}_t^{-1} \boldsymbol{C}_t, \end{aligned} \tag{51}$$

and solving for $\boldsymbol{\Sigma}_s$. The variance $\boldsymbol{\Sigma}_u$ of the control noise is then given by

$$\boldsymbol{\Sigma}_u = \boldsymbol{B}^{\dagger} \boldsymbol{\Sigma}_s \boldsymbol{B}^{\dagger T}. \tag{52}$$

The variance of our stochastic feedback controller does not depend on the controller gains and can be pre-computed before estimating the controller gains. If the computed desired control noise is smaller than the real control noise of the system, we use the control noise of the system to calculate the feedback gain matrix $\boldsymbol{K}$. Otherwise the estimated $\boldsymbol{\Sigma}_u$ is used to allow the trajectory distribution to increase its variance.

4.4.4 Controlling a physical system

On a non-linear physical system, we first obtain the linearization of the dynamics model using the current state $\boldsymbol{y}_t$ and use this linearization to obtain the parameters of the controller for the current time step in an online manner.

For a physical system, we also have to consider that the variance of the control noise $\boldsymbol{\Sigma}_{\boldsymbol{u}}$, computed from Eq. (52), contains two sources of noise; first, the inherent system noise $\boldsymbol{\Sigma}'_u$, and, second, the additional noise injected into the system by the demonstrator. Therefore, if we apply the control noise $\boldsymbol{\Sigma}_{\boldsymbol{u}}$ the inherent system noise will still be present and, as a result, our controller will not match the demonstrated distribution as it already contained the system noise. Therefore, we compute the control noise covariance

$$\boldsymbol{\Sigma}_{\boldsymbol{u}}^{[\mathrm{new}]} = \boldsymbol{\Sigma}_{\boldsymbol{u}} - \boldsymbol{\Sigma}'_u \tag{53}$$

by subtracting the estimated system noise $\boldsymbol{\Sigma}'_u$ from the controller noise $\boldsymbol{\Sigma}_{\boldsymbol{u}}$, computed from Eq. (52). If the resulting controller noise is not positive definite, e.g., when the system noise estimate is higher than the control noise, we set the control noise to zero.

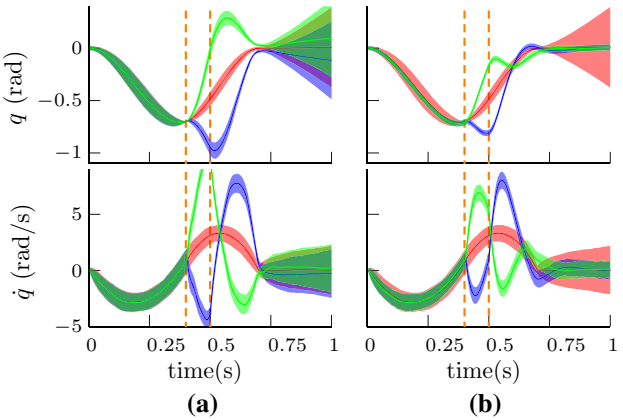


Fig. 8 *Robustness evaluation.* We applied a perturbation between the *dashed lines* with an amplitude of $P = 200\,(\mathrm{m/s^2})$ (*green*), or an amplitude of $P = -200\,(\mathrm{m/s^2})$ (*blue*). The ProMPs **a** show compliant behavior but pass through the via-point accurately. The DMPs **b** are much stiffer and compensate the perturbation faster, before the via-point was reached. The DMPs exhibit a less efficient recovery strategy due to the higher actions. **a** ProMPs, **b** DMPs (Color figure online)

Illustrative example—robustness analysis. In order to evaluate the robustness of our approach, we test different MP approaches under strong perturbation occurring during the execution of the movement, see Fig. 8. Our control approach demonstrates compliant behavior when the variance of the movement is high. It allows larger deviations from the demonstrated distribution and takes more time to "return" to the distribution. However, it manages to pass accurately through the via-points as this point has small variance. The DMPs on the other hand, use high feedback gains which results in a less compliant movement which quickly tries to return to the mean trajectory. Such strategy results in unnecessary high control actions as DMPs do not have a notion of the importance of time points.

4.4.5 Relation to optimal control

Our controller derivation has strong relations to optimal control (OC). Equation (42) resembles a continuous time Ricatti equation that is typically used for state estimation (Todorov 2008), only the observation noise is missing as it is not present in our application. It is well known that state estimation and optimal control are dual problems that can be solved in the same framework (Todorov 2008). Yet, our usage of the Ricatti equation is quite different from OC and state estimation. Both approaches use the Ricatti equation for backwards integration of the value function, or the covariance, respectively. In contrast, we assume that the covariance and its derivative are already known. In this case, we can use the Ricatti equation

Table 3 Overview of the experimental evaluation of ProMPs

Experiment	Real robot	#DoF	Basis Type	#Demos	#Basis	Evaluation objectives
7-link Reach.	Sim.	7	Gaussian	200	20	Movement coordination, via-points, combination
Double Pend.	Sim.	2	Gaussian	100	36	Non-linear system, change in the dynamics
Astrojax	✓	7	Von-Mises	7 periods	30	Periodic movements, movement coordination
Maracas	✓	7	Von-Mises	5 periods	10	Periodic movements, temporal modulation, blending
Hockey	✓	7	Gaussian	10 + 10	10	Union, combination, conditioning, context
Table Tennis	Sim.	7	Gaussian	20	15	Generalization in a complex noisy environment

to obtain the controller gains and no backwards integration is required. By circumventing the backwards integration, we can also avoid limitations of many OC algorithms. Almost all OC methods require a linearization of the model along a nominal mean trajectory. Using this linearization, an approximately optimal *linear* controller can be obtained (Li and Todorov 2010; Toussaint 2009). In contrast, our ProMP controller is non-linear as the linearization of the system is computed online for the current state. The use of OC or state estimation would also require that we know either the reward function or the observation model. Both quantities are unknown in the imitation learning scenario.

5 Experiments

We evaluate our approach on simulated and real robot experiments. Our experimental setups cover several aspects of our framework, i.e., stroke-based and rhythmic movements, linear and non-linear systems, simple trajectory following tasks, coordinated movements, and complex experiments such as table tennis or robot hockey.

For the real-robot experiments, i.e., the Astrojax, the maracas and the hockey task, we gathered demonstrations by kinesthetic teach-in, whereas for the simulated tasks we specify a cost function for finding the optimal time-varying controller. We used the optimal control algorithm from Toussaint (2009). For stroke-based movements, we train our approach as in Sect. 4.2.1 and for periodic tasks we use the EM approach in Sect. 4.2.2. An overview of the experiments performed and their objectives is given in Table 3. The open parameters of our approach where hand-picked and no further tuning was necessary.

5.1 7-link reaching task

In this task, we use a seven link planar robot that has to reach desired target positions in task-space, at different time points, with its end-effector. Our goal is to demonstrate the co-activation of ProMPs to solve a combination of tasks by combining two different movements. In addition, the task evaluates the necessity of the coupling between the joints of the robot, which is implemented by the ProMPs. As many joint configurations can lead to the same end-effector position, the end-effector of the robot can exhibit high accuracy, whereas each individual joint can exhibit higher variability. In this experiment, the end-effector has low variability at the task-space via-points. In order to successfully reproduce the demonstrated movements, ProMPs must correctly capture and reproduce the coupling between the DoF of the robot.

In the first set of demonstrations, the robot has to reach the via-point at $t_1 = 0.25$ s. The reproduced behavior with the ProMPs is illustrated in Fig. 9 (top). We learned the coupling of all seven joints with one ProMP. The ProMP exactly reproduced the via-points in task space while exhibiting a large variability for time steps in between the via-points. Moreover, the ProMP could also reproduce the coupling of the joints from the optimal control law which can be seen by the small variance of the end-effector in comparison to the rather large variance of the single joints at the via-points. The ProMP achieved an average cost value of similar quality as the optimal controller.

In the second set of demonstrations the first via-point was located at time step $t_2 = 0.75$ s. The movement of the robot is illustrated for specific time steps in Fig. 9 (middle). We combined both primitives and the resulting movement is illustrated in Fig. 9 (bottom). The combination of both MPs accurately reaches both via-points at $t_1 = 0.25$ and $t_2 = 0.75$, generating a primitive that satisfies *both* tasks.

Moreover, we evaluated the reproduction cost our approach to the number of training demonstrations in Fig. 10. The comparison was performed on the first set of demonstrations, i.e. top row of Fig. 9. With only two training demonstrations, our approach depends heavily on the regularization coefficients for the estimation of the covariance matrix and, on average, produces higher actions compared to using more demonstrations for training. In Fig. 10, we show that the performance of our approach does not significantly improve using more than 20 demonstrations for training. Additionally, we evaluated the performance of the inverse covariance controller (Calinon et al. 2010b) and the DMPs (Ijspeert et al. 2003). The cost for every experiment is averaged over 200 reproduc-

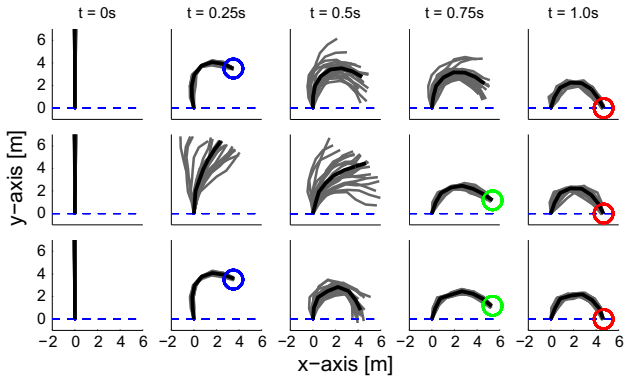


Fig. 9 A 7-link planar robot has to reach a target position at $T = 1.0$ s with its end-effector while passing a via-point at $t_1 = 0.25$ s (*top*) or $t_2 = 0.75$ s (*middle*). The plot shows the mean posture of the robot at different time steps in *black* and samples generated by the ProMP in *gray*. The ProMP approach was able to exactly reproduce the demonstration which have been generated by an optimal control law. The combination of both learned ProMPs is shown in the *bottom*. The resulting movement reached both via-points with high accuracy

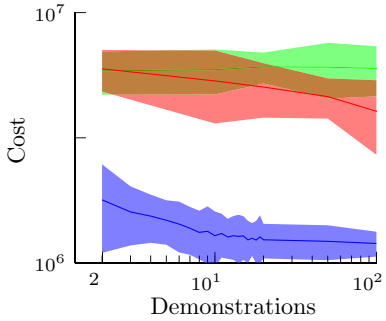


Fig. 10 Evaluation of the reproduction cost versus the number of demonstrations provided for training on the 7-link task-space via-point task. We present the results using ProMPs (*blue*), the Inv. Cov. Ctl. (*red*) Calinon et al. (2010b), and DMPs (green) Ijspeert et al. (2003). The cost is averaged over 200 reproductions for every approach and over 10 trials (Color figure online)

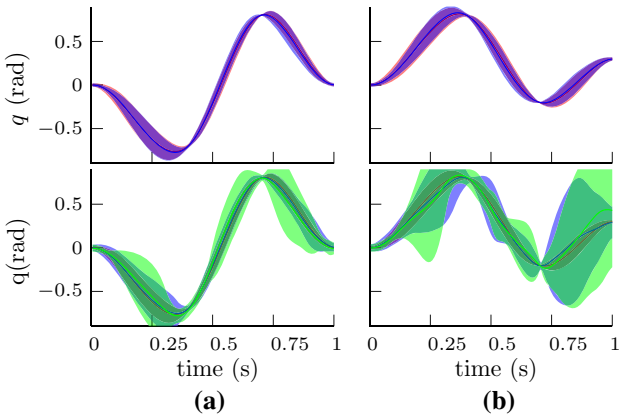


Fig. 11 *Double pendulum, non-linear system.* In *red* we depict the demonstrated trajectory distribution. (*first row*) In this experiment, we use the optimal controller to generate demonstrations on a linear system. Subsequently, we executed our controller on a non-linear double-pendulum system. The reproduced trajectory distribution (*blue*) match the demonstrations (*red*) despite the changed dynamics. The ProMP controller is using the linearization at the current state to compute the control gains. (*second row*) We illustrate the performance of our approach by using *non* state-independent gains (*blue*) where the linearization is performed offline along the mean state trajectory. As can be seen, ProMPs with state-independent gains are not capable of reproducing the demonstrated trajectory distribution. In *green*, we evaluate the performance of a linearized version of the non-linear ProMP controller which has been learned by fitting a linear model to the data produced by the ProMP controller. Also the linearized ProMP controller fails at tracking the distribution, showing that the state-dependent gains of the ProMP controller that cause the non-linearity are essential for accurate tracking in non-linear systems. **a** First joint and **b** second joint (Color figure online)

tions. Additionally, we average over 10 trials, where for each trial, we randomly randomly regenerated the demonstrations using an optimal control law.

5.2 Double pendulum

In this experiment we evaluate our control approach on a system with non-linear dynamics. We use a simulated double-pendulum with unit link lengths and unit masses. Non-linearities are induced due to gravity, centripetal and Coriolis forces. During the execution of our controller we compute a linearization of the system dynamics at every time step at the state $\boldsymbol{y}_t$ to obtain $\{\boldsymbol{A}_t, \boldsymbol{B}_t, \boldsymbol{c}_t\}$.

In this experiment, we also evaluate the robustness of the controller to changes in the system dynamics. To this end, we generated demonstrations on a linear double-link system, i.e. without gravity, centripetal, and Coriolis forces taken into account, using the optimal controller. Subsequently, we executed the learned trajectory distribution on the non-linear dynamical system using the ProMP controller that uses the linearization of the real dynamics. The linearization is performed in an online manner at the current state of the system for each of the reproductions, resulting in state-dependent gains and a non-linear control architecture. Our results are presented in Fig. 11. The reproduced trajectory distribution matches the demonstrations, despite the drastic change in the dynamics of the system. Additionally, we compare to the ProMP controller if we use a pre-linearization of the system dynamics along the mean trajectory, which is given in Fig. 11 (second row). Linearizing at the mean trajectory results in a linear feedback controller with state-independent gains and, hence, the resulting controller can not reproduce the demonstrated trajectory distribution. Moreover, we evaluated the reproduction a learned linear Gaussian controller per time-step which is learned from data obtained from the ProMP controller. We used the ProMP reproductions as our classical optimal control method (Toussaint 2009) failed to find a solution that was minimizing the given cost function. This approach is a linearized version of the non-linear

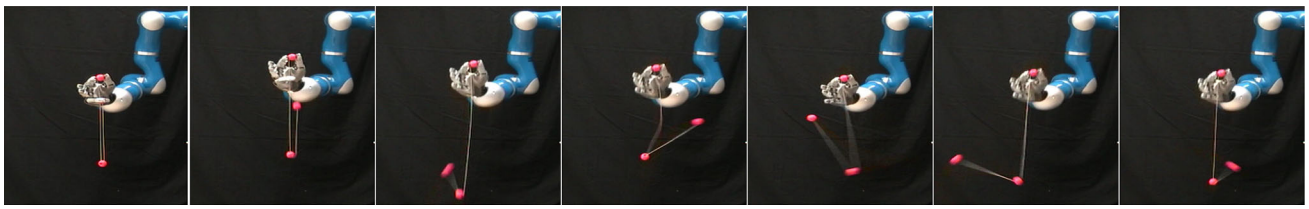


Fig. 12 The KUKA light-weight arm playing "Astrojax". The robot holds one of the balls in his fingers and starts with releasing the ball that is connected to the other end of the string. It subsequently reproduces the demonstrated rhythmic movement showing the same human-like variability in its movement pattern

ProMP controller. Our results in Fig. 11 show that the tracking performance reduces significantly, which proofs that the non-linearities of the ProMP controller are essential for accurate distribution tracking in non-linear systems.

5.3 Playing astrojax

'Astrojax' is a toy consisting of three balls on a string. Two balls are fixed at either end of the string, while one ball is free to slide along the string. Roughly, 'Astrojax' is a game between 'YoYo' and juggling. In order to successfully play 'Astrojax', the bottom two balls should orbit each other and not get in touch. We use the 'Astrojax' experiment to demonstrate that ProMPs can successfully learn and reproduce periodic movements. The real-robot setup is shown in in Figs. 1 and 12. The hand performs a stable grasp and is not controlled by ProMPs. We demonstrate a rhythmic movement to the robot which created a "basic orbit" pattern. We subsequently use the ProMPs to learn the movement with thirty Von-Mises basis functions for each joint. The robot could reproduce the behavior and recreated the same pattern, as illustrated in Fig. 12. The demonstrations exhibit a lot of variability and the robot generate periodic movements which show the same type of variability. During the demonstrations, we were capable of sustaining a successful orbit of the 'Astrojax' for a mean duration of $t_{\text{demo}} = 8.2\,(\text{s})$. During the reproduction, we achieved a mean orbiting of $t_{\text{reprod.}} = 15.2\,(\text{s})$. In contrast, the DMP approach would repeat always exactly the same movement, rendering the behavior different than the demonstrated one. DMPs are neither capable of reproducing variability, be compliant, or generate coordinated movements. GMR approaches, to our knowledge, have not yet investigated the application in periodic movements. A video with the robot playing 'Astrojax' can be found at http://www.ausy.tu-darmstadt.de/uploads/Team/AlexandrosParaschos/Astrojax.mp4.

5.4 Robot maracas

The maracas is a musical instrument containing grains. Shaking the maracas produces sounds. We used the KUKA lightweight arm for the experiments and the DLR hand to grasp the instrument. The hand was only used for holding the maracas and was not controlled by the ProMPs. Our setup is shown in Fig. 1.

As demonstrating fast movements with kinesthetic teach-in can be difficult on the real robot arm due to the inertia, friction, and model discrepancies, we demonstrate a slower movement of ten periods. We used this slow demonstration for learning the primitive but modulated the speed of the phase during reproduction. The faster movement achieved a shaking movement of appropriate speed that generates the desired sound of the instrument.

We learned the rhythmic movement using $N = 10$ Von-Misses basis functions per dimension. The ProMP was trained all seven DoF of the robot. We optimized the parameters of ProMPs using the Expectation Maximization algorithm. To do so, we split the demonstration in $M = 400$ segments and assigned the appropriate phase signal. We executed our controller after training and we measured that the generated trajectories stay on average 94.4% of the total time within two standard deviations of demonstrated distribution. After learning the ProMP model from the demonstration, we progressively increase the speed of the movement by modulating the phase, such that the robot successfully plays the instrument.

The speed of the motion can be changed during execution to achieve different sound patterns. We show an example movement of the robot in Fig. 13a. The desired trajectory distribution of the demonstrated rhythmic movement and the resulting distribution generated from the feedback controller again match.

Additionally, we demonstrated a second type of rhythmic shaking movement and use it to continuously blend between both movements to produce different sounds. One such transition between the two ProMPs is shown for one joint in Fig. 13b, c. We measured the trajectory reproduction accuracy from our controller against the desired blended distributions and found that the trajectories are within two standard deviations for 92.7, and 93.4% of the total execution time, respectively. A video showing the demonstration phase, reproduction with time modulation, and blending two primitives can be found at http://www.ausy.tu-darmstadt.de/uploads/Team/AlexandrosParaschos/Maracas.mp4

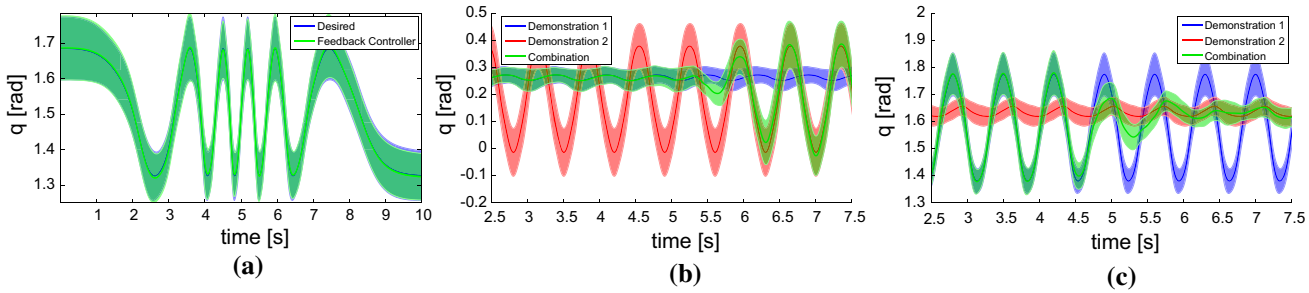


Fig. 13 **a** The trajectory distribution of the fourth joint when playing maracas. The speed of the movement is adapted by modulating the speed of the phase signal z_t. The desired distribution is shown in *blue* and the generated distribution from the feedback controller in *green*. Both distributions match. **b**, **c** Blending between two rhythmic movements (*blue* and *red* areas). The *green area* is produced by continuously switching from the *blue* to the *red* movement (Color figure online)

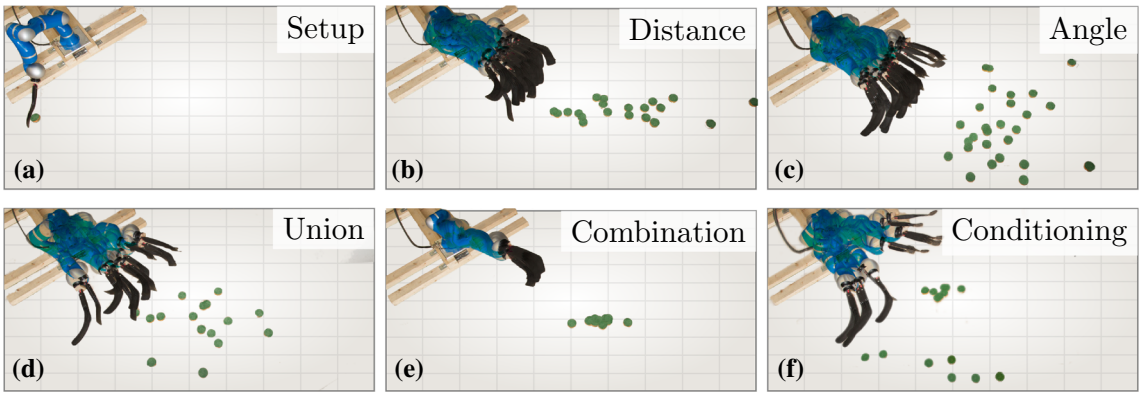


Fig. 14 Robot hockey. The robot shoots a hockey puck. The figure shows overlaid images of the real-robot setup that is set on the flour, taken from above. We demonstrate ten straight shots with varying distances and ten shots with varying angles. The pictures show samples from the ProMP model for straight shots (*b*) and shots with different angles (*c*). Learning from the union of the two data sets yields a model that represents variance in both distance and angle (*d*). Co-activating the individual MPs leads to a combined MP that reproduces shots where both models had probability mass, i.e., in the center at medium distance (*e*). The last picture shows the effect of conditioning on the angle of the shoot (*f*)

5.5 Robot hockey

In the hockey task, the robot has to shoot a hockey puck in different directions and for different distances. The task setup is depicted in Fig. 14a. We used the KUKA lightweight arm for this experiment and controlled the accelerations of the arm with the ProMPs using an inverse dynamics controller. The control parameters of the robot $t_{k\in 1\ldots K}$ are the desired position vector $\boldsymbol{q}_t \in \mathbb{R}^7$ and the desired acceleration $\ddot{\boldsymbol{q}}_t \in \mathbb{R}^7$ of each joint. The ProMPs provide at every time point the desired acceleration $\ddot{\boldsymbol{q}}_t$, while the desired position $\boldsymbol{q}_t$ is obtained from second-order Euler integration of the acceleration. The duration of the control step is dt $= 1$ ms. A hockey stick is mounted as an end-effector for hitting the puck.

We again used two sets of demonstrations. The first set contained $M_1 = 10$ demonstrations where the robot shot the puck straight at varying distances. The demonstrations were provided by a human tutor, using kinesthetic teaching. The second set also contained $M_2 = 10$ demonstrations where the demonstrator shot the puck at varying angles, while trying to keep the variance of the distance relatively small. For both demonstration sets, we trained two ProMPs using $N = 10$ Gaussian basis functions per dimension, which resulted in a weight vector $\boldsymbol{w} \in \mathbb{R}^{70}$. By reproducing the learned primitives, we obtain behaviors illustrated in Fig. 14b, c respectively. The shots exhibit the demonstrated variability in either angle or distance. We generated the images in Fig. 14 by taking the picture of the robot's configuration after the execution of the primitive and the puck has stopped. The figures show an overlay of the images from multiple executions of each primitive. By training a primitive on the union of the two datasets, the robot is able to shoot the puck at a variety of angles and distances, as illustrated in Fig. 14d. Additionally, we co-activated the two individual primitives and the resulting MP shoots only in the center

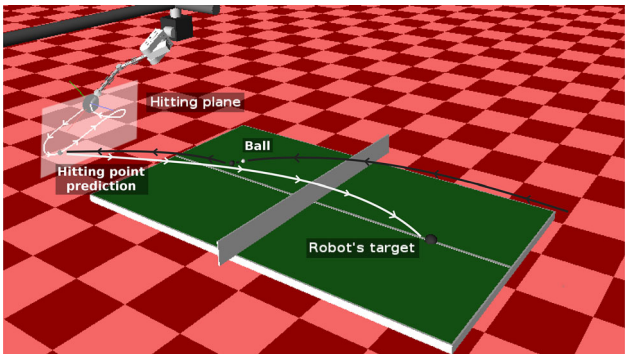


Fig. 15 The simulated table tennis setup. (*left*) Shown are the robot arm mounted on linear axis, the ball position, the hitting plane in which the robot will try to hit the ball, and the hitting point prediction. Due to the induced noise in our simulation the desired and actual hitting points may differ. On the opponent's side, we can see the robot's target for this simulation. In our experiments, we use 15 different combinations of initial ball positions and targets covering most of the table

Table 4 Evaluation of the average cost for the Robot Hockey experiment

Dataset	Average cost
Demonstrations	
Distance	1.20 ± 1.18
Angle	2.21 ± 2.95
Reproduction	
Distance	1.24 ± 1.24
Angle	2.07 ± 3.16
Combination	2.52 ± 1.59
Evaluation	
Dist. on Comb.	6.21 ± 8.18
Angle on Comb.	25.97 ± 21.54

We present the average cost of the human demonstrations for both demonstrated datasets. The robot reproduction results in similar cost as the demonstrations. The "Combination" cost is specified as the sum of both cost functions. The robot produces a novel composed behavior that performs significantly better than both demonstrated sets

at medium distance, i.e., the intersection of both MPs, as illustrated in Fig. 14e. This experiment again illustrates the achievement of a combination of tasks, where the first task was to shoot at a desired angle and the second, to shoot at a desired distance.

Finally, we learned a conditional distribution over the trajectories conditioned on the angle of the final puck position as described in Sect. 4.3.2. The resulting primitive was able to shoot at the desired angle as illustrated in Fig. 14f. All the operations are computed in closed form, no re-estimation of the primitive parameters is needed to compute the generalization or the combination of the primitives.

We provide a cost function evaluation of the two demonstrated datasets, the "angle" and the "distance" dataset, and the respective reproduction in Table 4. The cost function is chosen intuitively to resemble the desired task. By giving the human demonstrator a specific task, we can assume that he is minimizing a similar cost function, at least in approximation. Our approach successfully reproduces the same costs as in the demonstrations. The cost function of the "distance" dataset contains demonstrations that shoot the puck at different distances, but aiming at the same angle. Therefore, it only penalizes deviations from the desired angle. Similarly, in the "angle" dataset, the cost function penalizes deviations from the desired distance. Since, shooting the puck at a specific distance is quite hard due to different environment variables, i.e, friction between the puck surface and the floor, we choose a lower deviation penalty.

We also evaluated the cost on the combined movement which is supposed to solve both tasks, i.e., shoot at a specific distance and angle. For this evaluation, we added the cost functions from the "distance" and "angle" datasets. In Table 4, we show that the reproduction of the combination, which is a newly composed behavior not present in the demonstrations, achieves significantly lower costs than both original datasets.

5.6 Simulated table tennis

In this experiment, we evaluate the generalization capabilities of the ProMPs for a complex task. As comparison, we use the DMP approach presented in Kober et al. (2010). The robot, a simulated BioRob 5-DoF arm (Klug et al. 2008), is mounted on two linear axis and equipped with and additional shoulder joint. The setup is shown in Fig. 15. We control the robot with inverse dynamics control. We used an imperfect inverse dynamics model to render the simulation more realistic. As a result, the desired and actual trajectories do not match exactly and, thus, make the robot more sensitive to jerky movements as jerky movements are harder to track. At the beginning of each experiment, the ball is set to different pre-specified positions and initial velocities.

The robot has to return the ball to a specific target area at the opponents field. For this experiment, we gathered trajectories for 15 different combinations of initial ball configurations and robot targets, generated from an analytical player (Muelling et al. 2011). We trained the ProMP approach with the whole data-set and created a single primitive. In our experiment, the ball state is set at the beginning of a trial and the ProMP is conditioned to the predicted hitting position and velocity in joint space, obtained from the analytical player. A delay before the start of the execution of the primitive is provided by the simulation. In order to make the task more realistic, we assume that the ball state is estimated, instead of being directly observed, with zero-mean i.i.d. Gaussian noise. The noise on the ball position increases the task difficulty significantly as it also affects the estimated time until

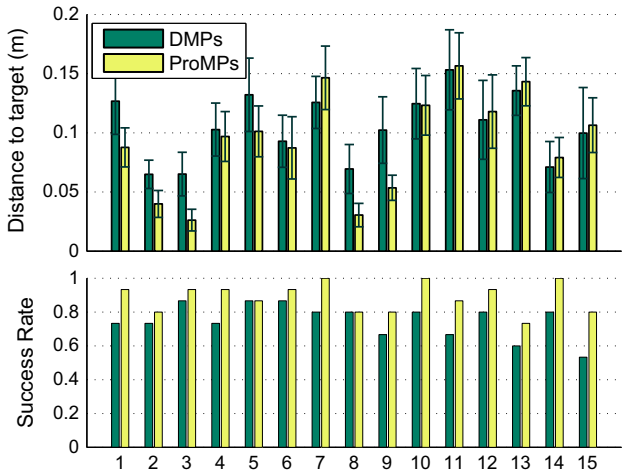


Fig. 16 The distance between the impact position of the ball on the opponents field and the actual targeted point in meters, for the DMP and the ProMP approaches. We tested 15 different configurations of ball initial states and robot's targets. We average the results over 20 samples where Gaussian observation noise was added to the initial ball position. The *bars* denote the mean error and the *error-bars* one standard deviation. (*bottom*) Shows the success rate for each combination. If the distance between the landed position and the target position is less than 0.4 meters it is counted as a success. The performance of ProMPs is superior in all the experiments leading generally to smaller errors with an increased success rate

the ball reaches the hitting plane. We evaluate the ProMPs and the DMPs on each of the 15 task setups by computing the average distance to the target and the average success rate. We display our results on Fig. 16.

The DMP was trained with only one demonstration, while the goal position and velocity were modified according to predicted hitting point using the approach presented in Kober et al. (2010). The DMP had inferior performance as it significantly deforms the trajectories, which makes the resulting trajectory harder to track as the feedback controller saturates in torque limits due the deformation. This saturation has the effect that the robot does not reach the specified hitting point with the specified velocity.

6 Discussion and conclusion

Probabilistic movement primitives are a promising approach for learning, modulating, and re-using movements in a modular control architecture. To effectively take advantage of such a control architecture, ProMPs support simultaneous activation, match the quality of the encoded behavior from the demonstrations, are able to adapt to different desired target positions, and can be efficiently learned by imitation. In ProMPs we parametrize the desired trajectory distribution of the primitive by a hierarchical Bayesian model with Gaussian distributions. The trajectory distribution can be easily obtained from demonstrations and simultaneously defines a feedback controller which is used for movement execution. Our probabilistic formulation introduces new operations for movement primitives, such us conditioning and combination of primitives. These all these mechanisms do not exist for alternative representations and, with ProMPs, we provide a single mathematical framework to describe them. Future work will focus on using the ProMPs in a modular control architecture and improving upon imitation learning by reinforcement learning.

The advanced flexibility of ProMPs comes to a cost of requiring multiple demonstrations in order to accurately encode the distribution over the trajectories. The number of demonstrations required depend on the complexity of the task and, from our experience, $\sim 10-20$ suffice for simple tasks. Prior knowledge about the task can be incorporate by using prior distributions and regularization techniques. Furthermore, our approach is appropriate for tasks that have a strong coupling to time. For tasks loosely coupled with time, other approached might produce better results. Finally, it should be noted that our approach can not capture multiple modes since we only use a single Gaussian component to encode the trajectory distribution.

Acknowledgements The research leading to these results has received funding from the European Community's Framework Programme CoDyCo (FP7-ICT-2011-9 Grant No. 600716), CompLACS (FP7-ICT-2009-6 Grant No. 270327), GeRT (FP7-ICT-2009-4 Grant No. 248273), and ERC StG SKILLS4ROBOTS.

References

Bruno, D., Calinon, S., Malekzadeh, M. S., & Caldwell, D. G. (2015). Learning the stiffness of a continuous soft manipulator from multiple demonstrations. In *Intelligent robotics and applications* (pp. 185–195).

Buchli, J., Stulp, F., Theodorou, E., & Schaal, S. (2011). Learning variable impedance control. *International Journal of Robotics Research*, *30*(7), 820–833.

Calinon, S. (2016). A tutorial on task-parameterized movement learning and retrieval. *Intelligent Service Robotics*, *9*(1), 1–29.

Calinon, S., D'Halluin, F., Sauser, E. L., Caldwell, D. G., & Billard, A. G. (2010). Learning and reproduction of gestures by imitation. *IEEE Robotics and Automation Magazine*, *17*, 44–54.

Calinon, S., Sardellitti, I., & Caldwell, D. G. (2010b). Learning-based control strategy for safe human–robot interaction exploiting task and robot redundancies. In *IEEE/RSJ international conference on intelligent robots and systems (IROS)* (pp. 249–254).

Daniel, C., Neumann, G., & Peters, J. (2012). Learning concurrent motor skills in versatile solution spaces. In *IEEE/RSJ international conference on intelligent robots and systems (IROS)*, (pp. 3591–3597).

da Silva, B., Konidaris, G., & Barto, A. (2012). Learning parameterized skills. In *International conference on machine learning* (pp. 1679–1686).

dAvella, A., & Bizzi, E. (2005). Shared and specific muscle synergies in natural motor behaviors. *Proceedings of the National Academy of Sciences (PNAS)*, *102*(3), 3076–3081.

Degallier, S., Righetti, L., Gay, S., & Ijspeert, A. (2011). Toward simple control for complex, autonomous robotic applications: Combining discrete and rhythmic motor primitives. *Autonomous Robots*, *31*, 155–181.

Dominici, N., Ivanenko, Y. P., Cappellini, G., dAvella, A., Mondì, V., Cicchese, M., et al. (2011). Locomotor primitives in newborn babies and their development. *Science*, *334*(6058), 997–999.

Ernesti, J., Righetti, L., Do, M., Asfour, T., & Schaal, S. (2012). Encoding of periodic and their transient motions by a single dynamic movement primitive. In *IEEE-RAS international conference on humanoid robots (humanoids)* (pp. 57–64).

Ewerton, M., Maeda, G., Peters, J., & Neumann, G. (2015). Learning motor skills from partially observed movements executed at different speeds. In *IEEE/RSJ international conference on intelligent robots and systems (IROS)* (pp. 456–463).

Forte, D., Gams, A., Morimoto, J., & Ude, A. (2012). On-line motion synthesis and adaptation using a trajectory database. *Robotics and Autonomous Systems*, *60*, 1327–1339.

Gams, A., Nemec, B., Ijspeert, A. J., & Ude, A. (2014). Coupling movement primitives: Interaction with the environment and bimanual tasks. *IEEE Transactions on Robotics*, *30*(4), 816–830.

Higham, N. J. (1988). Computing a nearest symmetric positive semidefinite matrix. *Linear Algebra and its Applications*, *103*, 103–118.

Ijspeert, A. J. (2008). Central pattern generators for locomotion control in animals and robots: A review. *Neural Networks*, *21*(4), 642–653.

Ijspeert, A. J., Nakanishi, J., Hoffmann, H., Pastor, P., & Schaal, S. (2013). Dynamical movement primitives: Learning attractor models for motor behaviors. *Neural Computation*, *25*(2), 328–373.

Ijspeert, A. J., Nakanishi, J., & Schaal, S. (2003). Learning attractor landscapes for learning motor primitives. In *Advances in neural information processing systems (NIPS)* (pp. 1547–1554).

Khansari-Zadeh, S. M., & Billard, A. (2011). Learning stable nonlinear dynamical systems with Gaussian mixture models. *IEEE Transactions on Robotics*, *27*(5), 943–957.

Khansari-Zadeh, S. M., Kronander, K., & Billard, A. (2014). Modeling robot discrete movements with state-varying stiffness and damping: A framework for integrated motion generation and impedance control. In *Robotics science and systems (R:SS)*.

Klug, S., Lens, T., von Stryk, O., Möhl, B., & Karguth, A. (2008). Biologically inspired robot manipulator for new applications in automation engineering. In *Proceedings of robotik*.

Kober, J., Muelling, K., Kroemer, O., Lampert, C. H., Scholkopf, B., & Peters, J. (2010). Movement templates for learning of hitting and batting. In *International conference on robotics and automation (ICRA)* (pp. 853–858).

Konidaris, G., Kuindersma, S., Grupen, R., & Barto, A. (2012). Robot learning from demonstration by constructing skill trees. *International Journal of Robotics Research (IJRR)*, *31*(3), 360–375.

Kormushev, P., Calinon, S., & Caldwell, D. G. (2010). Robot motor skill coordination with EM-based reinforcement learning. In *International conference on intelligent robots and systems (IROS)* (pp. 3232–3237).

Kulvicius, T., Ning, K., Tamosiunaite, M., & Worgotter, F. (2012). Joining movement sequences: Modified dynamic movement primitives for robotics applications exemplified on handwriting. *IEEE Transactions on Robotics*, *28*(1), 145–157.

Lazaric, A., & Ghavamzadeh, M. (2010). Bayesian multi-task reinforcement learning. In *International conference on machine learning (ICML)* (pp. 599–606).

Li, W., & Todorov, E. (2010). Iterative linear quadratic regulator design for nonlinear biological movement systems. In *International conference on informatics in control, automation and robotics (ICINCO)* (pp. 222–229).

Maeda, G., Ewerton, M., Lioutikov, R., Amor, H., Peters, J., & Neumann, G. (2014). Learning interaction for collaborative tasks with probabilistic movement primitives. In *International conference on humanoid robots (Humanoids)* (pp. 527–534).

Matsubara, T., Hyon, S. H., & Morimoto, J. (2011). Learning parametric dynamic movement primitives from multiple demonstrations. *Neural Networks*, *24*(5), 493–500.

Moro, F. L., Tsagarakis, N. G., & Caldwell, D. G. (2012). On the kinematic motion primitives (kMPs)—Theory and application. *Frontiers in Neurorobotics*, *6*(10), 1–18.

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

Mülling, K., Kober, J., Kroemer, O., & Peters, J. (2013). Learning to select and generalize striking movements in robot table tennis. *The International Journal of Robotics Research*, *32*(3), 263–279.

Nakanishi, J., Morimoto, J., Endo, G., Cheng, G., Schaal, S., & Kawato, M. (2004). Learning from demonstration and adaptation of biped locomotion. *Robotics and Autonomous Systems*, *47*, 79–91.

Neumann, G., Daniel, C., Paraschos, A., Kupcsik, A., & Peters, J. (2014). Learning modular policies for robotics. *Frontiers in Computational Neuroscience*, *8*(62), 1.

Neumann, G., Maass, W., & Peters, J. (2009). Learning complex motions by sequencing simpler motion templates. In *International conference on machine learning (ICML)* (pp. 753–760)

OHagan, A., & Forster, J. (2004). Kendalls advanced theory of statistics: Bayesian inference (2nd ed.). Arnold, New York. Technical report, ISBN 0-340-80752-0.

Paraschos, A., Daniel, C., Peters, J., & Neumann, G. (2013a). Probabilistic movement primitives. In *Advances in neural information processing systems (NIPS)* (pp. 2616–2624).

Paraschos, A., Neumann, G., & Peters, J. (2013b). A probabilistic approach to robot trajectory generation. In *International conference on humanoid robots (humanoids)* (pp. 477–483)

Pastor, P., Hoffmann, H., Asfour, T., & Schaal, S. (2009). Learning and generalization of motor skills by learning from demonstration. In *International conference on robotics and automation (ICRA)* (pp. 763–768)

Pastor, P., Righetti, L., Kalakrishnan, M., & Schaal, S. (2011). Online movement adaptation based on previous sensor experiences. In *International conference on intelligent robots and systems (IROS)* (pp. 365–371)

Peters, J., Mistry, M., Udwadia, F. E., Nakanishi, J., & Schaal, S. (2008). A unifying methodology for robot control with redundant DOFs. *Autonomous Robots*, *24*(1), 1–12.

Righetti, L., & Ijspeert, A. J. (2006). Programmable central pattern generators: An application to biped locomotion control. In *International conference on robotics and automation, (ICRA)* (pp. 1585–1590).

Rozo, L., Calinon, S., Caldwell, D., Jiménez, P., & Torras, C. (2013). Learning collaborative impedance-based robot behaviors. In *AAAI conference on artificial intelligence* (pp. 1422–1428).

Rückert, E. A., Neumann, G., Toussaint, M., & Maass, W. (2012). Learned graphical models for probabilistic planning provide a new class of movement primitives. *Frontiers in Computational Neuroscience*, *6*(97), 1.

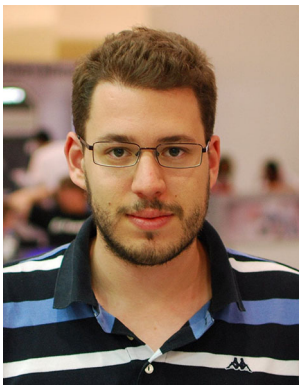
Rueckert, E., Mundo, J., Paraschos, A., Peters, J., & Neumann, G. (2015). Extracting low-dimensional control variables for movement primitives. In *International conference on robotics and automation (ICRA)* (pp. 1511–1518).

Schaal, S., Mohajerian, P., & Ijspeert, A. (2007). Dynamics systems vs. optimal control—A unifying view. *Computational Neuroscience: Theoretical Insights into Brain Function*, *165*, 425–445.

Schaal, S., Peters, J., Nakanishi, J., & Ijspeert, A. (2005). Learning movement primitives. In *International symposium on robotics research* (pp. 561–572).

Stark, H., & Woods, J. (2001). *Probability and random processes with applications to signal processing* (3rd ed.). Upper Saddle River: Prentice-Hall.

Stengel, R. F. (2012). *Optimal control and estimation*. North Chelmsford, MA: Courier Corporation.
Todorov, E. (2008). General duality between optimal control and estimation. *Conference on Decision and Control*, *5*, 4286–4292.
Todorov, E., & Jordan, M. (2002). Optimal feedback control as a theory of motor coordination. *Nature Neuroscience*, *5*, 1226–1235.
Toussaint, M. (2009). Robot trajectory optimization using approximate inference. In *International conference on machine learning (ICML)* (pp. 1049–1056).
Ude, A., Gams, A., Asfour, T., & Morimoto, J. (2010). Task-specific generalization of discrete and periodic dynamic movement primitives. *Transactions in Robotics*, *5*, 800–815.
Williams B., Toussaint, M., & Storkey, A. (2007). Modelling motion primitives and their timing in biologically executed movements. In *Advances in neural information processing systems (NIPS)* (pp. 1609–1616).



Alexandros Paraschos received his PhD from Technische Universität Darmstadt. Previously, he studied Electronic and Computer Engineering at Technical University of Crete and worked at Cognitive Robotics Research Centre (CRRC), at University of Wales, Newport, as a research associate. He specializes in movement representation for motor skills in redundant robots. He aims to create the movement representation that will allow robots to share our environment, but until then he focuses on creating movement representations that not only allow composing complex robot skills out of elemental movements, but also have extensive generalization capabilities.

Christian Daniel is a research scientist at the Bosch Center for Artificial Intelligence. Previously, he received his Ph.D. from TU Darmstadt's Intelligent Autonomous System lab. His research interest include machine learning and reinforcement learning for physical systems.

Jan Peters is a full professor (W3) for Intelligent Autonomous Systems at the Computer Science Department of the Technische Universitaet Darmstadt and at the same time a senior research scientist and group leader at the Max-Planck Institute for Intelligent Systems, where he heads the interdepartmental Robot Learning Group. He has received the Dick Volz Best 2007 US PhD Thesis Runner-Up Award, the Robotics: Science & Systems - Early Career Spotlight, the INNS Young Investigator Award, and the IEEE Robotics & Automation Society's Early Career Award. Recently, he received an ERC Starting Grant.

Gerhard Neumann is a Professor of Robotics & Autonomous Systems in College of Science. Before coming to Lincoln, he has been an Assistant Professor at the TU Darmstadt from September 2014 to October 2016 and head of the Computational Learning for Autonomous Systems (CLAS) group. Before that, he was Post-Doc and Group Leader at the Intelligent Autonomous Systems Group (IAS) also in Darmstadt under the guidance of Prof. Jan Peters. He obtained his Ph.D. under the supervision of Prof. Wolfgang Mass at the Graz University of Technology. He already authored 50+ peer reviewed papers, many of them in top ranked machine learning and robotics journals or conferences such as NIPS, ICML, ICRA, IROS, JMLR, Machine Learning and AURO. In Darmstadt, he is principle investigator of the EU H2020 project Romans and also already acquired DFG funding. He organized several workshops and is senior program committee for several conferences.