To appear in *Advanced Robotics*
Vol. 00, No. 00, Month 20XX, 1–18

FULL PAPER

Probabilistic Movement Primitives under Unknown System Dynamics

Alexandros Paraschos[a]*, Elmar Rueckert[a], Jan Peters[a,c], and Gerhard Neumann[d]

[a] *Intelligent Autonomous Systems, TU Darmstadt, Darmstadt, Germany*
{rueckert}@ias.tu-darmstadt.de;
[b] *Robot Learning Group, Max Planck Institute for Intelligent Systems, Germany*
mail@jan-peters.net;
[c] *Lincoln Centre for Autonomous Systems (LCAS), University of Lincoln, UK*
gneumann@lincoln.ac.uk

(*Received 00 Month 20XX; accepted 00 Month 20XX*)

Physical interaction requires robots to accurately follow kinematic trajectories while modulating the interaction forces to accomplish tasks and to be safe to the environment. However, current approaches rely on accurate physical models or iterative learning approaches. We present a versatile approach for physical interaction tasks, based on Movement Primitives (MPs) that can learn physical interaction tasks solely by demonstrations, without explicitly modelling the robot or the environment. We base our approach on the Probabilistic Movement Primitives (ProMPs), which utilizes the variance of the demonstrations to provide better generalization of the encoded skill, combine skills, and derive a controller that follows exactly the encoded trajectory distribution. However, the ProMP controller requires the system dynamics to be known. We present a reformulation of the ProMPs that allows accurate reproduction of the skill without modelling the system dynamics and, further, we extent our approach to incorporate external sensors, as for example, force/torque sensors. Our approach learns physical interaction tasks solely from demonstrations and online adapts the movement to force-torque sensor input. We derive a variable-stiffness controller in closed form that reproduces the trajectory distribution and the interaction forces present in the demonstrations. We evaluate our approach in simulated and real-robot tasks.

1. Introduction

Developing robots that operate in the same environment with humans and physically interacting with everyday objects requires accurate control of the contact forces that occur during the interaction. While non-compliant robots can achieve a great accuracy, the uncertainty of complex and less-structured environment prohibits physical interaction. In this paper, we focus on providing a compliant control scheme that can enable robots to manipulate their environment, in a safe manner, without damaging it, and enabling human-robot collaboration. Force-control approaches have been successfully used in such scenarios, e.g. robots dancing in collaboration with humans [1], or turning a rope [2, 3]. Typically, force-control requires an accurate dynamics model of the robot and its environment that is not easy to obtain. Other approaches suggest to learn a dynamics model, however, this process can be time-consuming and is prone to model-errors. We present an approach that can jointly learn the desired movement of the robot and the contact forces by human demonstrations, without relying on a learned forward or inverse model.

Existing approaches for motor skill learning that are based on movement primitives [4–8], often incorporate into the movement primitive representation the forces needed for the physical interactions [9–11]. However, such approaches model a single successful reproduction of the task. Multiple demonstrations are typically averaged, despite that they actually represent similar, but different, solutions of the task. Thus, the applied contact forces are not correlated with the state of the robot nor sensory values that indicate the state of the environment, e.g., how heavy an object is.

We propose learning the coordination of the interaction forces, with the kinematic state of the system, as

well as the control actions needed to reproduce the movement exclusively from demonstration. Motor skill learning for such interaction tasks for high-dimensional redundant robots is challenging. This task requires real-time feedback control laws that process sensory data including joint encoders, tactile feedback and force-torque readings. We present a model-free version of the Probabilistic Movement Primitives (ProMPs) [12] that enables robots to acquire complex motor skills from demonstrations, while it can coordinate the movement with force, torque, or tactile sensing. The ProMPs have several beneficial properties, such us generalization to novel situations, combination of primitives and time-scaling, which we inherit in our approach.

ProMPs assume a locally linearizable dynamics models to compute time-varying feedback control laws. However, such dynamics models are hard to obtain for physical interaction tasks. Therefore we obtain a time varying feedback controller directly from the demonstration without requiring such a model. In the model-free extension of the ProMPs, we condition the joint distribution over states and controls on the current state of the system, and obtain a distribution over the controls. We show that this distribution represents a time-varying stochastic linear feedback controller. Due to the time-varying feedback gains, the controller can exhibit behavior with variable stiffness and, thus, it is safe to use in physical interaction. A similar control approach has recently been presented in [13], with the difference of requiring an additional optimization step, e.g., for avoiding joint limits, due to the movement representation used.

Our approach inherits many beneficial properties of the original ProMP formulation. We can reproduce the variability in the demonstrations and use probabilistic operators for generalization to new tasks or the co-activation of learned primitives. The resulting feedback controller shows similar properties as in the model-based ProMP approach, it can reproduce optimal behavior for stochastic systems and exactly follow the learned trajectory distribution, at least, if the real system dynamics are approximately linear for each time step. For non-linear systems, the estimated variable stiffness controller can get unstable if the robot reaches configurations that are far away from the set of demonstrations. To avoid this problem, we smoothly switch between a stable PD-controller and the ProMP controller if the support of the learned distribution for the current situation is small. We show that this extension allows us to track trajectory distributions accurately even for non-linear systems.

In this article, we extend our approach [14] to provide a more detailed explanation on sensory integration, introduce a mixture model of primitives, present how our approach can be used for adapting the interaction forces to user's input, and evaluate our approach in more complex robotic tasks.

2. Related Work

In this section, we review related work on movement primitives for imitation learning that combine position and force tracking, model the coupling between kinematics and forces and are able to capture the correlations between these two quantities.

The benefit of an additional feedback controller to track desired reference forces was demonstrated in grasping tasks in [9]. Individual dynamical systems (DMPs) [8] were trained for both, position and force profiles in imitation learning. The force feedback controller substantially improved the success rate of grasps in tracking demonstrated contact forces under changing conditions. For manipulation tasks like opening a door, the authors showed that the learned force profiles can be further improved through reinforcement learning [10]. However, the approach requires the manual tuning of the gains and has limited generalization capabilities of the movement [15]. This approach is applicable for tasks where learning a single reference force profile suffices and generalization to new situations assume that the system dynamics stay constant.

For many tasks, such as like bi-manual manipulations, the feedback controller needs to be coupled. Gams et al. [16] proposed *cooperative* dynamical systems, where deviations from desired forces modulate the velocity forcing term in the DMPs for position control. This approach was tested on two independently operating robot arms solving cooperative tasks like lifting a stick [11]. Deviations in the sensed contact forces in one robot were used to adapt the DMP of the other robot and the coupling parameters were ob-

tained through iterative learning control. A related probabilistic imitation learning approach to capture the couplings in time was proposed in [17]. In this approach, Gaussian mixture models were used to represent the variance of the demonstrations. For training this approach the robot first reproduces the learned positional movement and then, with the help of an external force input device, the force-profile is learned. The position and force profiles are coupled only in time and cross-correlations are not captured. The approach was evaluated successfully on complex physical interaction tasks such as ironing, opening a door, or pushing against a wall.

Adapting Gaussian Mixture Models (GMMs) [18–21] have been proposed for use in physical interaction tasks. The major difference to the dynamical systems approach is that GMMs can represent the variance of the movement. Closely related to our approach, Evrard et al. in [22] used GMMs to learn joint distributions of positions and forces. Joint distributions capture the correlations between positions and forces and were used to improve adaptation to perturbations in cooperative human robot tasks for object lifting. In this approach, the control gains were fixed to track the mean of the demonstrated trajectories. In [23], it was shown that by assuming known forward dynamics, variable stiffness control gains can be derived in closed form to match the demonstrations. We address here an important related question of how these gains can be learned in a model-free approach from the demonstrations.

3. Model-Free Probabilistic Movement Primitives

We propose a novel framework for robot control which can be employed in physical interaction scenarios. In our approach, we jointly learn the desired trajectory distribution of the robot's joints or end-effectors and the corresponding controls signals. We train our approach from a limited set of demonstrations. We refer to the joint distribution as state-action distribution. Further, we incorporate proprioceptive sensing, such as force or tactile sensing, into our state representation. The additional sensing capabilities are of high importance for physical interaction as they can disambiguate kinetically similar states. We present our approach by, first, extending the Probabilistic Movement Primitives (ProMPs) framework [12] to encode the state-action distribution and, second, we derive a stochastic feedback controller without the use of a given system dynamics model. Finally, we extend our control approach for states which are relatively far from the vicinity of the learned state-action distribution. In that case, our control approach can no longer produce correcting actions and an additional backup controller with high gains is needed. Our framework inherits most of the beneficial properties introduced by the ProMPs that significantly improved generalization to novel situations and enables the generation of primitives that *concurrently* solve multiple tasks [12].

3.1. *Encoding the Time-Varying State-Action Distribution of the Movement*

We avoid explicitly learning robot and environment models by learning directly the appropriate control inputs, while keeping the beneficial properties of the ProMP approach, such us generalization and concurrent execution.

In order to simplify the illustration of our approach, we first discuss the special case of a single Degree of Freedom (DoF) and, subsequently, we expand our description to the generic case of multiple DoF. The description is based in [12], but modified appropriately to clarify how the actions can be modelled. First, we define the extended state of the system as

$$\boldsymbol{y}_t = [q_t, \dot{q}_t, u_t]^T, \tag{1}$$

where q_t is the position of the joint, $\dot{q}_t$ the velocity, and u_t the control applied at time-step t. Similar to ProMPs, we use a linear basis function model to encode the trajectory of the extended state $\boldsymbol{y}_t$. The feature

matrix and the weight vector of the non-linear function approximation model become

$$\boldsymbol{y}_t = \begin{bmatrix} q_t \\ \dot{q}_t \\ u_t \end{bmatrix} = \tilde{\boldsymbol{\Phi}}_t \boldsymbol{w}, \quad \tilde{\boldsymbol{\Phi}}_t = \begin{bmatrix} \boldsymbol{\phi}_t^T & \mathbf{0} \\ \dot{\boldsymbol{\phi}}_t^T & \mathbf{0} \\ \mathbf{0} & \boldsymbol{\psi}_t^T \end{bmatrix}, \boldsymbol{w} = \begin{bmatrix} \boldsymbol{w}_q \\ \boldsymbol{w}_u \end{bmatrix}, \tag{2}$$

where the vectors $\boldsymbol{\phi}_t$ and $\boldsymbol{\psi}_t$ represent the feature vectors for the position q_t and the control u_t respectively. The derivative of the position feature vector $\dot{\boldsymbol{\phi}}_t$ is used to compute the velocity of the joint $\dot{q}_t$. The weight vector $\boldsymbol{w}$ contains the weight vector for the position $\boldsymbol{w}_q$ and the weight vector for the control $\boldsymbol{w}_u$. The dimensionality of the feature $\boldsymbol{\phi}_t$ and weight $\boldsymbol{w}_q$ vectors is $N \times 1$, where N is the number of features used to encode the joint position. Similarly, the dimensionality of $\boldsymbol{\psi}_t$ and $\boldsymbol{w}_u$ vectors is $M \times 1$. The remaining entries of $\tilde{\boldsymbol{\Phi}}_t$, denoted by $\mathbf{0}$, are zero-matrices with the appropriate dimensionality. In our approach, we distinct between the features used to encode the position from the features used to encode the control signal due to the different properties of the two signals. The distinction allows us to use of different type of basis functions, different parameters, or a different number of basis functions.

We extend our description to the multidimensional case. First, we extend the state of the system from Equation (2) to

$$\boldsymbol{y}_t = \left[\boldsymbol{q}_t^T, \dot{\boldsymbol{q}}_t^T, \boldsymbol{u}_t^T\right]^T, \tag{3}$$

where the vector $\boldsymbol{q}_t$ is a concatenation of the positions of all joints of the robot, the vector $\dot{\boldsymbol{q}}_t$ of the velocities of the joints, and $\boldsymbol{u}_t$ of the controls respectively. The feature matrix $\tilde{\boldsymbol{\Phi}}_t$ now becomes a block matrix

$$\tilde{\boldsymbol{\Phi}}_t = \left[\boldsymbol{\Phi}_t^T, \dot{\boldsymbol{\Phi}}_t^T, \boldsymbol{\Psi}_t^T\right]^T, \tag{4}$$

where

$$\boldsymbol{\Phi}_t = \left[\begin{array}{ccc|c} \boldsymbol{\phi}_t^T & \cdots & \mathbf{0} & \\ \vdots & \ddots & \vdots & \mathbf{0} \\ \mathbf{0} & \cdots & \boldsymbol{\phi}_t^T & \end{array}\right], \boldsymbol{\Psi}_t = \left[\begin{array}{c|ccc} & \boldsymbol{\psi}_t^T & \cdots & \mathbf{0} \\ \mathbf{0} & \vdots & \ddots & \vdots \\ & \mathbf{0} & \cdots & \boldsymbol{\psi}_t^T \end{array}\right], \tag{5}$$

define the features for the joint positions and the joint controls. Similarly to the single DoF, the features used for the joint velocities $\dot{\boldsymbol{\Phi}}_t$ are the time derivatives of the features of the joint positions $\boldsymbol{\Phi}_t$. We use the same features for every DoF. The dimensionality of the feature matrices $\boldsymbol{\Phi}_t$ and $\boldsymbol{\Psi}_t$ is $K \times K \cdot (N + M)$, where K denotes the number of DoF.

The weight vector $\boldsymbol{w}$ has a similar structure to Equation (2) and, for the multi-DoF case, is given by

$$\boldsymbol{w} = \Big[\underbrace{{}^1\boldsymbol{w}_q^T, \cdots, {}^K\boldsymbol{w}_q^T}_{\text{weights for joint positions}}, \underbrace{{}^1\boldsymbol{w}_u^T, \cdots, {}^K\boldsymbol{w}_u^T}_{\text{weights for joint controls}} \Big]^T, \tag{6}$$

where ${}^i\boldsymbol{w}$ denotes the weight vector for joint $i \in [1, K]$.

The probability of a single trajectory $\boldsymbol{\tau} = \{\boldsymbol{y}_t, t \in [1 \cdots T]\}$, composed from states of T subsequent time steps, given the parameters $\boldsymbol{w}$, is computed by

$$p(\boldsymbol{\tau}|\boldsymbol{w}) = \prod_t \mathcal{N}\left(\boldsymbol{y}_t | \boldsymbol{\Phi}_t \boldsymbol{w}, \boldsymbol{\Sigma}_{\boldsymbol{y}}\right), \tag{7}$$

where we assume *i.i.d.* Gaussian observation noise with zero mean and $\boldsymbol{\Sigma}_{\boldsymbol{y}}$ covariance. Representing multiple trajectories would require a set of weights $\{\boldsymbol{w}\}$. Instead of explicitly maintaining such a set, we

introduce a distribution over the weights $p(\boldsymbol{w};\boldsymbol{\theta})$, where the parameter vector $\boldsymbol{\theta}$ defines the parameters of the distribution. Given the distribution parameters $\boldsymbol{\theta}$, the probability of the trajectory becomes

$$p(\boldsymbol{\tau};\boldsymbol{\theta}) = \int p(\boldsymbol{\tau}|\boldsymbol{w})p(\boldsymbol{w};\boldsymbol{\theta})\mathrm{d}\boldsymbol{w}, \tag{8}$$

where we marginalize over the weights $\boldsymbol{w}$. As in the ProMP approach, we use a Gaussian distribution to represent $p(\boldsymbol{w};\boldsymbol{\theta})$, where $\boldsymbol{\theta} = \{\boldsymbol{\mu}_{\boldsymbol{w}}, \boldsymbol{\Sigma}_{\boldsymbol{w}}\}$. Using a Gaussian distribution enables the marginal to be computed analytically and facilitates learning. The distribution over the weight vector $p(\boldsymbol{w};\boldsymbol{\theta})$ correlates (couples) the DoFs of the robot to the action vector at every time-step t. The probability of the current state-action vector $\boldsymbol{y}_t$ given $\boldsymbol{\theta}$ is computed by

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

in closed form. We use normalized Gaussian basis functions as features. Each basis function is defined in the time domain by

$$\phi_i(t) = \frac{b_i(t)}{\sum_{j=1}^{n} b_j(t)}, \quad b_i(t) = \exp\left(-\frac{(t-c_i)^2}{2h}\right), \tag{9}$$

where c_i denotes the center of the ith basis function and h the bandwidth. The centers of the basis functions are spread uniformly in $[-2h, T_{\text{end}} + 2h]$. The number of basis functions and the bandwidth value we used, depend on the complexity of task. Typically, complex task require higher number of basis functions in order to represent them accurately.

3.2. *Imitation Learning for Model-Free ProMPs*

We use multiple demonstrations to estimate the parameters $\boldsymbol{\theta} = \{\boldsymbol{\mu}_{\boldsymbol{w}}, \boldsymbol{\Sigma}_{\boldsymbol{w}}\}$ of the distribution over the weights $p(\boldsymbol{w}|\boldsymbol{\theta})$. First, for each demonstration i, we use linear ridge regression to estimate the parameter vector $\boldsymbol{w}_i$ associated to that specific demonstration, i.e.,

$$\boldsymbol{w}_i = (\boldsymbol{\Phi}_t^T\boldsymbol{\Phi}_t + \lambda\boldsymbol{I})^{-1}\boldsymbol{\Phi}_t^T\boldsymbol{Y}_i, \tag{10}$$

where λ denotes the ridge factor and $\boldsymbol{Y}_i$ the observations of the state and action for all the time steps of that demonstration. We set λ to zero, unless numerical issues arise. Subsequently, we estimate the parameters $\boldsymbol{\theta}$ from the set of weights $\{\boldsymbol{w}_i, i \in [1, N]\}$ using the ML estimators for Gaussians, i.e.,

$$\boldsymbol{\mu}_{\boldsymbol{w}} = \frac{1}{L}\sum_{i=1}^{L}\boldsymbol{w}_i, \quad \boldsymbol{\Sigma}_{\boldsymbol{w}} = \frac{1}{L}\sum_{i=1}^{L}(\boldsymbol{w}_i - \boldsymbol{\mu}_{\boldsymbol{w}})(\boldsymbol{w}_i - \boldsymbol{\mu}_{\boldsymbol{w}})^T, \tag{11}$$

where L is the number of demonstrations.

3.3. *Integration of Proprioceptive Feedback*

Additional sensory feedback integration, e.g., force-torque feedback, is beneficial for physical interaction scenarios, as we demonstrate in Section 4.5, because our approach can capture the correlation of the trajectory, the controls and the sensory signal. This correlation contains useful information for the reproduction of the movement.

We extend our approach to additionally contain the sensory signal $\boldsymbol{s}_t$ in the state $\boldsymbol{y}_t$, i.e., Equation (3) becomes

$$\boldsymbol{y}_t = \left[\boldsymbol{q}_t^T, \dot{\boldsymbol{q}}_t^T, \boldsymbol{s}_t, \boldsymbol{u}_t^T\right]^T. \tag{12}$$

The derivative of the sensory signal $\dot{\boldsymbol{s}}_t$ is typically not included in the state. The respective basis function matrix becomes

$$\tilde{\boldsymbol{\Phi}}_t = \left[\boldsymbol{\Phi}_t^T, \dot{\boldsymbol{\Phi}}_t^T, \boldsymbol{Z}_t^T, \boldsymbol{\Psi}_t^T\right]^T, \tag{13}$$

where $\boldsymbol{Z}_t^T$ denotes the basis functions used for the external sensory signal $\boldsymbol{s}_t$. In this paper we set the basis function $\boldsymbol{Z}_t^T$ similarly to $\boldsymbol{\Phi}_t^T$, but we provide a generic derivation to allow the use of other basis functions. We train our approach as shown in Section 3.2, solely from demonstrations. The weight vector $\boldsymbol{w}_s$ is now expanded to accommodate the weights for the additional sensory feedback dimensions, i.e.,

$$\boldsymbol{w} = \Big[\underbrace{{}^1\boldsymbol{w}_q^T, \cdots, {}^K\boldsymbol{w}_q^T}_{\text{weights for joint positions}}, \quad \underbrace{{}^1\boldsymbol{w}_s^T, \cdots, {}^K\boldsymbol{w}_s^T}_{\text{weights for force/torque sensors}}, \underbrace{{}^1\boldsymbol{w}_u^T, \cdots, {}^K\boldsymbol{w}_u^T}_{\text{weights for joint controls}} \Big]^T, \tag{14}$$

hence, by learning the distribution $p(\boldsymbol{w})$, we can represent the correlations between the sensory signal and the control commands. We use the sensory signal to get a new desired trajectory distribution and its controls.

3.4. *Generalization with Conditioning*

The modulation of via-points and final positions is an important property of any MP framework to adapt to new situations. Generalization to different via-points or final targets can be implemented by conditioning the distribution at reaching the desired position $\boldsymbol{q}_t^*$ at time step t.

By applying Bayes theorem, we obtain a new distribution $p(\boldsymbol{w}|\boldsymbol{q}_t^*)$ for $\boldsymbol{w}$ which is Gaussian with mean and variance

$$\boldsymbol{\mu}_{\boldsymbol{w}}^{[\text{new}]} = \boldsymbol{\mu}_{\boldsymbol{w}} + \boldsymbol{Q}_t \left(\boldsymbol{q}_t^* - \boldsymbol{\Psi}_t^T \boldsymbol{\mu}_{\boldsymbol{w}}\right), \tag{15}$$

$$\boldsymbol{\Sigma}_{\boldsymbol{w}}^{[\text{new}]} = \boldsymbol{\Sigma}_{\boldsymbol{w}} - \boldsymbol{Q}_t \boldsymbol{\Psi}_t^T \boldsymbol{\Sigma}_{\boldsymbol{w}}, \tag{16}$$

$$\boldsymbol{Q}_t = \boldsymbol{\Sigma}_{\boldsymbol{w}} \boldsymbol{\Psi}_t \left(\boldsymbol{\Sigma}_q^* + \boldsymbol{\Psi}_t^T \boldsymbol{\Sigma}_{\boldsymbol{w}} \boldsymbol{\Psi}_t\right)^{-1}, \tag{17}$$

where $\boldsymbol{\Sigma}_q^*$ is a covariance matrix specifying the accuracy of the conditioning. By conditioning to the desired position, the weight vectors for the controls $\boldsymbol{w}_{\boldsymbol{u}}$ are modulated as well. Therefore, the proposed controller of Section 3.5 will drive the system to the desired state $\boldsymbol{q}_t^*$. The interaction forces can be adapted in our approach, if it is physically possible, using conditioning in a similar fashion. To this end, the user has to specify the desired force or torque, i.e., $\boldsymbol{s}_t^*$ and accuracy $\boldsymbol{\Sigma}_{\boldsymbol{s}}^*$. We evaluate the conditioning operator in Section 4.1 for conditioning the to desired positions and in Section 4.3.

3.5. *Robot Control with Model-Free ProMPs*

We derive a stochastic feedback controller which is ideally capable of reproducing the learned distribution. We define as $\tilde{\boldsymbol{y}}_t$ the observable state of the system, that contains the joint positions, velocities, and

potentially force or torque data, but not the action. We rewrite the joint probability

$$p(\boldsymbol{y}_t) = p(\tilde{\boldsymbol{y}}_t, \boldsymbol{u}_t) = \mathcal{N}\left(\begin{bmatrix} \tilde{\boldsymbol{y}}_t \\ \boldsymbol{u}_t \end{bmatrix} \middle| \tilde{\boldsymbol{\Phi}}_t \boldsymbol{\mu}_{\boldsymbol{w}}, \tilde{\boldsymbol{\Phi}}_t \boldsymbol{\Sigma}_{\boldsymbol{w}} \tilde{\boldsymbol{\Phi}}_t^T + \boldsymbol{\Sigma}_{\boldsymbol{y}} \right),$$

where

$$\tilde{\boldsymbol{\Phi}}_t \boldsymbol{\Sigma}_{\boldsymbol{w}} \tilde{\boldsymbol{\Phi}}_t^T = \begin{bmatrix} \boldsymbol{\Phi}_t \boldsymbol{\Sigma}_{\boldsymbol{w}} \boldsymbol{\Phi}_t^T & \boldsymbol{\Phi}_t \boldsymbol{\Sigma}_{\boldsymbol{w}} \boldsymbol{\Psi}_t^T \\ \boldsymbol{\Psi}_t \boldsymbol{\Sigma}_{\boldsymbol{w}} \boldsymbol{\Phi}_t^T & \boldsymbol{\Psi}_t \boldsymbol{\Sigma}_{\boldsymbol{w}} \boldsymbol{\Psi}_t^T \end{bmatrix}, \tag{18}$$

and condition on the current observable state $\tilde{\boldsymbol{y}}_t$ to obtain the desired action. From the Bayes theorem, we obtain the probability of the desired action

$$p(\boldsymbol{u}_t | \tilde{\boldsymbol{y}}_t) = \frac{p(\tilde{\boldsymbol{y}}_t, \boldsymbol{u}_t)}{p(\tilde{\boldsymbol{y}}_t)} = \mathcal{N}\left(\boldsymbol{u}_t | \boldsymbol{\mu}_{\boldsymbol{u}}, \boldsymbol{\Sigma}_{\boldsymbol{u}}\right), \tag{19}$$

which is a Gaussian distribution as both $p(\tilde{\boldsymbol{y}}_t)$ and $p(\boldsymbol{u}_t)$ are Gaussian. The mean and covariance of $p(\boldsymbol{u}_t)$ are computed by

$$\boldsymbol{\mu}_{\boldsymbol{u}} = \boldsymbol{\Psi}_t \boldsymbol{\mu}_{\boldsymbol{w}} + \boldsymbol{K}_t, (\tilde{\boldsymbol{y}}_t - \boldsymbol{\Phi}_t \boldsymbol{\mu}_{\boldsymbol{w}}) \tag{20}$$

$$\boldsymbol{\Sigma}_{\boldsymbol{u}} = \boldsymbol{\Psi}_t \boldsymbol{\Sigma}_{\boldsymbol{w}} \boldsymbol{\Psi}_t^T + \boldsymbol{K}_t \, \boldsymbol{\Phi}_t \boldsymbol{\Sigma}_{\boldsymbol{w}} \boldsymbol{\Phi}_t^T, \tag{21}$$

$$\boldsymbol{K}_t = \boldsymbol{\Psi}_t \boldsymbol{\Sigma}_{\boldsymbol{w}} \boldsymbol{\Phi}_t^T \left(\boldsymbol{\Phi}_t \boldsymbol{\Sigma}_{\boldsymbol{w}} \boldsymbol{\Phi}_t^T \right)^{-1}, \tag{22}$$

using Gaussian identities. We rewrite the mean control given the observable state $\tilde{\boldsymbol{y}}_t$ as

$$\boldsymbol{\mu}_{\boldsymbol{u}} = \boldsymbol{\Psi}_t \boldsymbol{\mu}_{\boldsymbol{w}} + \boldsymbol{K}_t \tilde{\boldsymbol{y}}_t - \boldsymbol{K}_t \boldsymbol{\Phi}_t \boldsymbol{\mu}_{\boldsymbol{w}} = \boldsymbol{K}_t \tilde{\boldsymbol{y}}_t + \boldsymbol{k}_t,$$

and observe that it has the same structure as a feedback controller with time varying gains. The feedback gain matrix $\boldsymbol{K}_t$ couples the DoF and the additional force-torque signals of the system. The control covariance matrix $\boldsymbol{\Sigma}_{\boldsymbol{u}}$ introduces correlated noise in the controls. The noise used only if we want to match the variability of the demonstrations. Alternatively, we can disable the noise and replay the noise-free behavior.

3.6. *Correction Terms for Non-Linear Systems*

A basic assumption for the linear feedback controller obtained by the ProMP approach is that the movement is defined in a local vicinity such that a linear controller is sufficient. Whenever the robot's state "leaves" this vicinity, due to the non-linearities of the dynamics, the learned feedback controller might not be able to direct the robot back to the desired trajectory distribution. Therefore, we apply a correction controller that is active only when the state is sufficiently "far" outside the distribution and directs the system to the mean of the demonstrated state distribution. The correction controller is defined as a standard PD controller with hand-tuned gains, i.e.,

$$\boldsymbol{u}_t^C = \boldsymbol{K}_P \left(\boldsymbol{\mu}_{q,t} - \boldsymbol{q}_t \right) + \boldsymbol{K}_D \left(\boldsymbol{\mu}_{\dot{q},t} - \dot{\boldsymbol{q}} \right) + \boldsymbol{u}_{\text{ff},t}, \tag{23}$$

where the feed forward term $\boldsymbol{u}_{\text{ff},t}$ is still estimated from the ProMP and given by the mean action of the ProMP for time step t, i.e.,

$$\boldsymbol{u}_{\text{ff},t} = \boldsymbol{K}_t \boldsymbol{\Phi}_t \boldsymbol{\mu}_{\boldsymbol{w}} + \boldsymbol{k}_t. \tag{24}$$

The correcting action $\boldsymbol{u}_t^C$ is only applied if we are outside the given trajectory distribution. We use a sigmoid activation function that depends on the log-likelihood of the current state to switch between the ProMP feedback controller and the correction controller,

$$\sigma\left(\boldsymbol{q}_t, \dot{\boldsymbol{q}}_t\right) = \frac{1}{1 + \exp\left(-\log\left(p\left(\boldsymbol{q}_t, \dot{\boldsymbol{q}}_t; \boldsymbol{\theta}\right)\right)\beta^{-1} - \alpha\right)}, \tag{25}$$

where α and β are hand tuned parameters of the activation function. We linearly interpolate between the controls of the ProMP and the correction action. For a high likelihood, e.g., $\sigma(\boldsymbol{q}_t, \dot{\boldsymbol{q}}_t) = 1$ we fully activate the feedback controller from the ProMP. For $\sigma(\boldsymbol{q}_t, \dot{\boldsymbol{q}}_t) = 0$ we fully activate the correction action.

3.7. *Time adaptation and mixture of primitives*

In this section, we extend our approach to naturally combine multiple primitives, necessary in tasks that the interaction with an object does not always occur at a specific point in time, or not occur at all. For example, when pushing a box, the robot would have to modulate its controller if it is in contact with the box. Placing the box in different locations and training our approach as presented in Section 3.2, will create a primitive that averages over both cases, contact and no-contact, and the robot will fail to reproduce the task.

Therefore, we introduce a primitive mixture model that allow the robot to automatically select the best primitive to execute, according to the sensory input and its position. Solely selecting a primitive would have limited use without being able to locally adjust the time to match, for example, exactly the time of contact with the object.

To compensate for time offsets in the movement, we substitute the explicit time relationship in our representation, with a function of time

$$z(t) = t + b, \tag{26}$$

which we define as phase. To incorporated the phase variable in Equation (3), we modify the basis functions $\boldsymbol{\Phi}_t$ to depend on the phase instead of explicitly in time. Since the phase is a function of time we keep the same notation for clearness. During learning, we estimate a probability distribution over the offset parameter b,

$$p(b; \boldsymbol{\theta}_b) \sim \mathcal{N}\left(b|\boldsymbol{\mu}_b, \boldsymbol{\Sigma}_b\right) \tag{27}$$

by fitting a Gaussian distribution. The probability distribution over the state of the system $\boldsymbol{y}_t$ in now given by

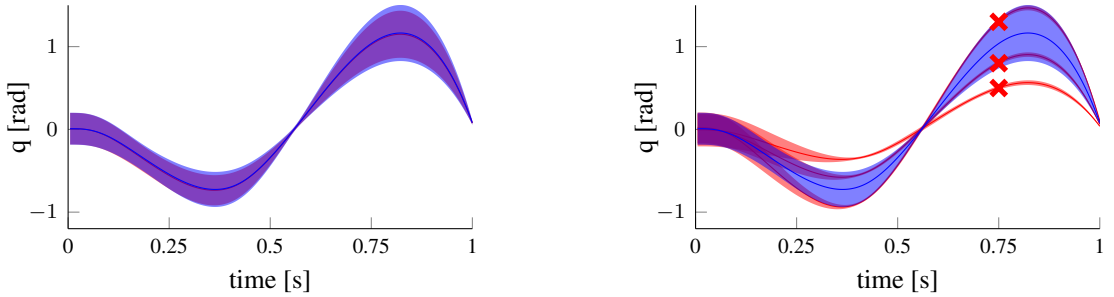
$$p(\boldsymbol{y}_t; \boldsymbol{\theta}, \boldsymbol{\theta}_b) = \int p(\boldsymbol{y}_t|b; \boldsymbol{\theta}) p(b; \boldsymbol{\theta}_b) \mathrm{d}b \approx \sum_{i=1}^{L} p(\boldsymbol{y}_t|b_i; \boldsymbol{\theta}), \tag{28}$$

which is approximated with L samples as it can not be computed analytically. The samples are begin drawn from the prior $p(b; \boldsymbol{\theta}_b)$. The probability of the state given the weight parameters and the sampled offset b_i, $p(\boldsymbol{y}_t|b_i; \boldsymbol{\theta})$, can be computed in closed form from Equation (8).

During reproduction, we select the primitive and offset sample that result in the highest likelihood, i.e.,

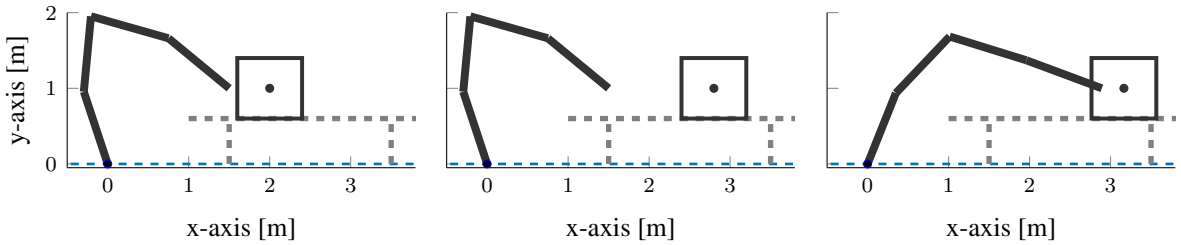
$$b^*, \boldsymbol{\theta}^* \approx \arg\max_{b, \boldsymbol{\theta}} p(\boldsymbol{y}_t|b_i; \boldsymbol{\theta}_j), \tag{29}$$

where $\boldsymbol{\theta}^*$ denote the parameters of the most suitable primitive and b^* the time offset.



(a) Reproduction (b) Generalization

Figure 1.: (a) We evaluate our approach on a simulated 1-DoF linear system. We use $N = 30$ demonstrations (red) for training. During the reproduction (blue) our approach matches exactly the demonstrations. (b) We evaluate the generalization capabilities of our approach with *conditioning*. The initial distribution is depicted in blue. At time $t = 0.75s$ we condition the initial distribution to pass at a specific position $q = \{0.5, 0.8, 1.3\}$ with low variance. We generate $N = 30$ demonstrations for every conditioning point and we show the resulting distribution in red. The *X* markers denotes the position at the conditioning point.



(a) Initial position, box placed close (b) Initial position, box placed far (c) Final position

Figure 2.: The setup of the box the generalization to different initial positions. We use a quad-link, joint control robot with one meter links to push with its end-effector a box to the final configuration, shown in (c). We demonstrated two primitives with the box placed to different initial positions, one within the proximity of the robot's end-effector (a) and one where the box was placed further away (b).

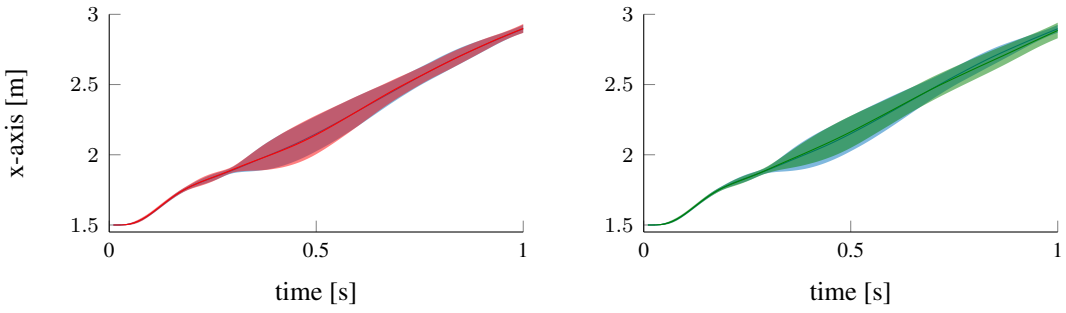
4. Experimental Evaluation

We begin the experimental evaluation on illustrative examples to demonstrate the properties of our approach. We start with a linear one dimensional system to demonstrate the accurate reproduction and generalization of the learned trajectory distribution and we proceed by applying our approach to a more complex system, a quad-link pendulum with non-linear dynamics.

Further, we perform evaluations on complex real-robot platforms. The humanoid robot iCub lifts a grate to a predetermined height from different grasping locations, without learning a model of the grate. Subsequently, we evaluate our approach on moving a chemistry flask of an unknown weight to a target location, while avoiding obstacles, using the KUKA LWR robotic arm.

4.1. *Reproduction and Generalization of the Trajectory Distribution*

In this section, we evaluate the approach on learning a trajectory distribution from demonstrations, generalizing the distribution to novel locations, and reproducing the learned distribution using our proposed control approach. For this evaluation, we used an one dimensional, linear, system with second order integrator dynamics. The demonstrations, used for illustrative purposes, were generated using fifth order splines to reach different via-points, followed by PD control law. We injected noise in the acceleration of the system.



(a) Reproduction (b) Generalization to random box positions

Figure 3.: The end-effector trajectory for pushing a box placed randomly on the table. In (a), the robot can reproduce exactely the demonstrations. In (b), we demonstrate that the robot reproduces the same end-effector trajectory, for any position between the two demonstrations.

The resulting trajectory distribution is shown in Fig. 1a (red). In the same figure, we illustrate the resulting trajectory distribution by using our proposed control approach in blue. Our proposed controller matches the demonstrated distribution accurately.

Further, the adaptation capabilities of our approach are evaluated using the conditioning operation that adapts the trajectory distribution to novel situations. We conditioned the trajectory distribution to reach positions $0.5, 0.8$ and 1.3 at time point $t = 0.75s$. Our proposed control approach manages to reach the desired position on every case as shown in Fig. 1b. The desired positions are indicated by red crosses. Our approach maintains the shape of the distribution and reaches the desired position with minimal deviations.

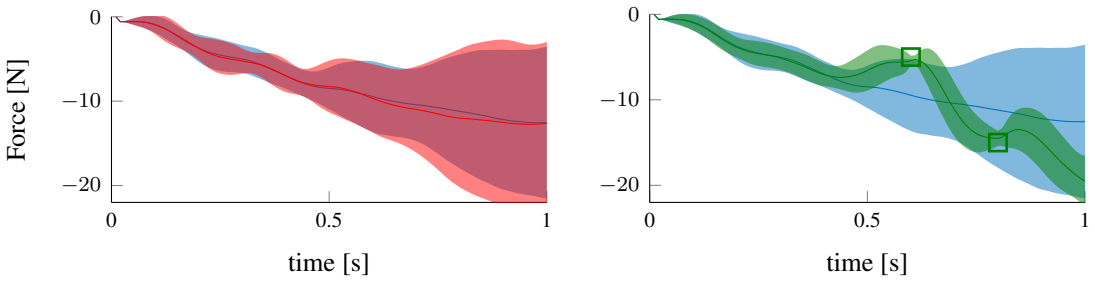
4.2. *Generalization to different initial positions using a mixture of primitives*

In this section, we present an evaluation of our approach using the proposed mixture of primitives to accommodate accurate reproduction when the initial position, and, hence, the time the end-effector of the robot makes contact with the object is unknown. In this demonstration, we used a quad link robot pushing a box placed on top of a table. The position of the box varies and is not observable from our approach. We trained our probabilistic model using three sets of twenty demonstrations, one set were the box's initial position on the x-axis was set to 2m, a set were the box was placed at 3m, and a primitive where there was no box present. In all three primitives the end-effector was following the same trajectory distribution. The demonstrations were created using a hand-tuned controller. In this evaluation, the robot is joint controlled for both the demonstrations and the reproduction. We simulate the interaction between the robot's end-effector and the box using a compliant spring-damper model. Hence, the robot's end-effector can slightly penetrate the box. The experimental setup is depicted in Fig. 2.

First, we evaluate our approach on the same scenarios as the demonstrations. We present our results for all three primitives in Fig.3a. The robot's end-effector using our proposed control approach follows the same trajectory distribution (red) as the demonstrations (blue). Additionally, we evaluate our approach using random initial positions for the box in the range 1m to 2m. The robot can reproduce successfully the movement for all the positions in this range. The resulting end-effector trajectory distribution for twenty reproductions is presented in Fig. 3b.

4.3. *Adaptation of the interaction forces*

In this section, we evaluate our approach on adapting the interaction forces during the execution of the learned primitive, to the desired values. We use the same setup as in Sec. 4.2, however, for this evaluation we set the mass of the box high enough for the robot not to be able to push it. We generated demonstration using a hand-tuned controller and we used our approach to reproduce the learned primitive. The robot



(a) Reproduction of the interaction forces (b) Adaptation of the interaction forces

Figure 4.: In this figure, we show the interaction forces between the robot's end-effector and a heavy box. In (a), the robot exerts the same force profile (red) as in the demonstrations (blue). In (b), the robot reproduces the adapted force profile (green). The green boxes illustrate the adaptation points.

replicates the same interaction force distribution as observed during the demonstrations. We present our results in Fig. 4a.

To evaluate the adaptation capabilities of the interaction forces, we conditioned the learned primitive to apply -5N at time $t = 0.6$s and -15N at $t = 0.8$s. During the execution of the primitive, the robot successfully reproduce the desired forces at the corresponding time, while during for the remaining time the interaction forces generated using the proposed controller were close to the demonstrations. We show our results in Fig. 4b.

4.4. *Non-Linear Quad-Link Pendulum*

To evaluate the quality of our controller on a non-linear system, we tested our model-free ProMP approach on a non-linear quad-link planar pendulum. Each link had a mass of 1kg and a length of 1m. We used the standard rigid body dynamics equations, where the gravity and the Coriolis forces are the major non-linear terms. We collected demonstrations by defining the desired trajectory as a spline with two via-point at $t = 0.3, 0.8$ in the task-space of the robot. We generated the demonstration trajectories using inverse kinematics for generating the joint space reference trajectories. Then, we used a inverse dynamics controller to track the reference trajectories and we collected the joint state-action data. We trained our approach using $N = 30$ demonstrations.

The resulting trajectory distribution for the y-dimension of the task-space is show in Fig. 5. The robot can track with its end-effector the desired distribution accurately and can reproduce the two via-points. In Fig. 6 we show all four the joint trajectories. In the joint space distributions the via-points are not visible but are captured in the covariance matrix of the weights. The controller could match the mean and variance of the demonstrated trajectory distribution. In Fig. 7, we illustrated the resulting trajectory from the controller in the task space of the robot. The activation of the correcting controller is around 1% of the total execution time.

4.5. *Adaptation to External Forces on the iCub*

In this experiment we used the presented model-free ProMP approach to learn a one-dimensional torque feedback controller in the humanoid robot iCub. The task is to tilt a grate multiple times from an initial distribution to a goal distribution, as show in Fig. 9a. In our experiments we use the wrist joint. The grate is attached to the robot at different lengths, to simulate different grasping locations. We demonstrate 20 movements per grasping location to train our approach. The data where recorded through tele-operation. In this experiment the state encodes the joint angle encoder value and the joint torque reading in the wrist. We present the recored torques from the sensor of the robot for all three demonstrated grasping locations in Fig. 9b. By placing the grate on the same location as during the demonstration and reproducing the

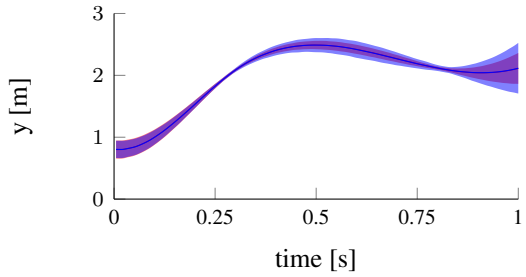


Figure 5.: We evaluate our approach on an non-linear system with $D = 4$ DoF. While the dynamics of the task are non-linear we are able to reproduce (blue) accurately the demonstrated distribution (red). We show the trajectory distribution of the "y" dimension of the task-space of the robot. Our approach captures the correlations between the DoF of the robot and reduces the variance of the trajectory reproduction at both via-points.

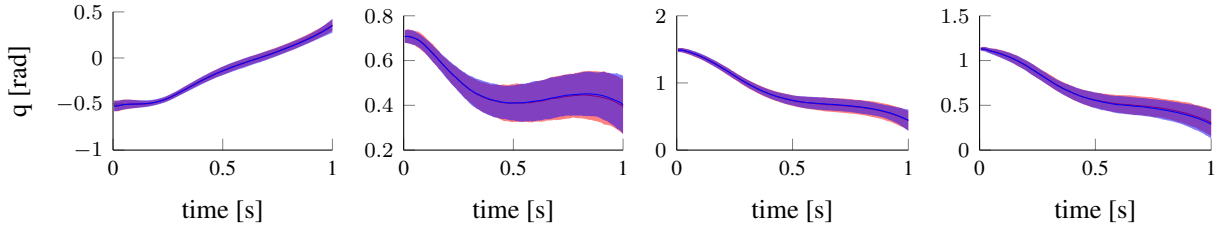


Figure 6.: The evaluation of our approach on the quad-link robot. We present the results of the of the DoF in joint space. The demonstrated distribution is plotted in red and the reproduction in blue. The two distributions match. The two via-points of the movement, which were set in task-space, are not visible in joint-space.

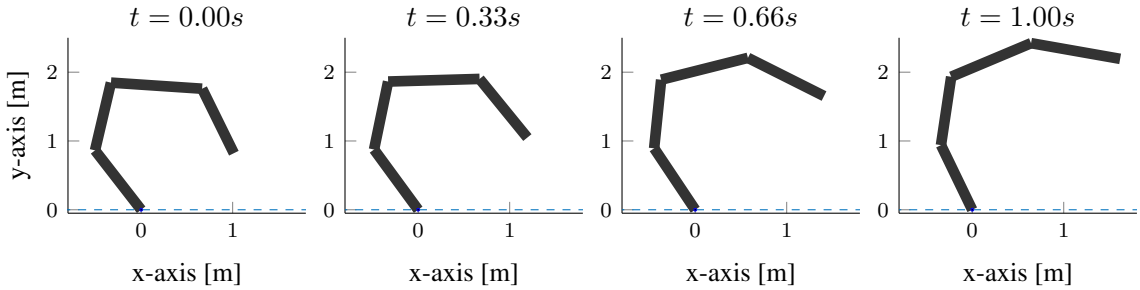


Figure 7.: An animation of the movement of the quad-link non-linear robot during the execution of our approach. We use darker colors at the beginning of the movement and lighter at the end.

movement with our approach, we show that we observe the same torque profile. The force measurement is crucial in our experiment as it is used for applying the correct forces during the execution of the movement. When disabled, the robot either fails to lift the grate to the demonstrated location or it overshoots. The overshooting is due to gravity, as in that grasping location the center of the mass of the grate is moved over the axis of wrist rotation. The results are shown in Fig. 10. The reproduction distributions where created using twenty executions of the model-free ProMP controller per grasping location. Our approach can generalize to different grasping locations between the demonstrations. We generalized into four new locations and executed our controller. The robot reproduces the same joint distribution while compensating for the different dynamics, as shown in Fig.9a.

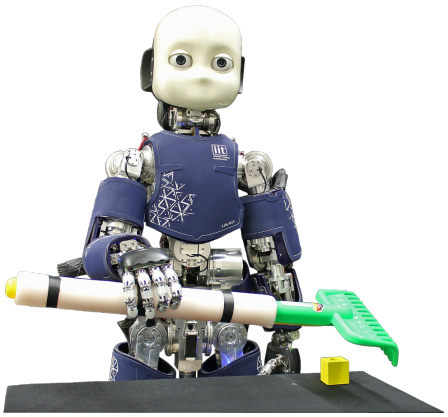
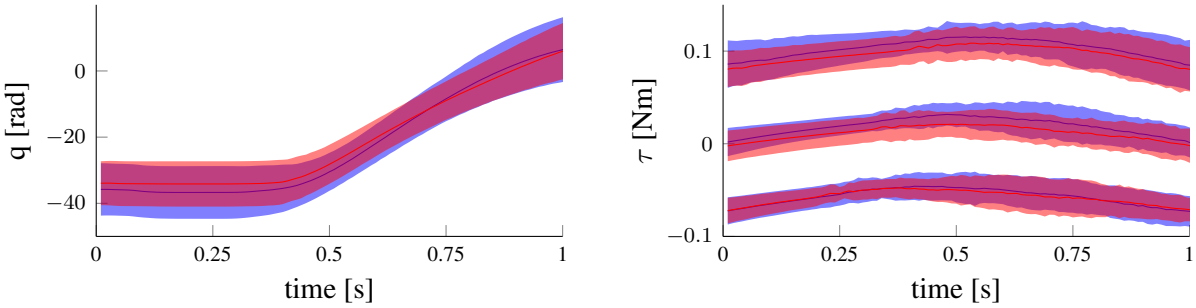


Figure 8.: The *iCub* robot is taught by imitation how to tilt a grate that we use of during the experimental evaluation of our approach. We demonstrated how to lift a grate from three different positions. Grasping from different positions change the dynamics of the task. Our method provides online adaptation and generalizes in the area of the grasps



(a) Trajectory distribution

(b) Torque distribution

Figure 9.: (a) The trajectory distribution of the wrist joint of the iCub during our experiment. The demonstrated distribution is presented in blue and the reproduction in red. The demonstrated distribution contain trajectories from all three grasping locations. The reproduction distribution contain trajectories from seven grasping locations. The Model-Free ProMPs can reproduce the demonstrated distribution in new grasping locations. (b) The torque distribution of all grasping locations used during the demonstrations. Each location created a distinct offset in the measured torque. We present the demonstrated torque distributions in blue. Additionally, we show that our approach can reproduce the torque distribution when we position the grate at the same locations as in the demonstrations. We present the reproduction results in red.

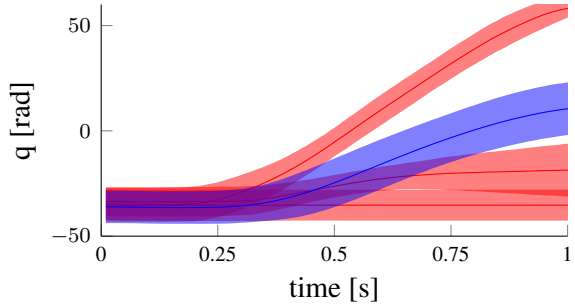


Figure 10.: The trajectory distribution of the wrist joint of the robot, when we disable the torque feedback. Depending on the grasping location, the robot either fails to lift the grate to the same height as demonstrated, or, it overshoots the lifting task due to gravity. In the later case, it should be noted that the center of mass of the grate is moved over the axis of the joint and, thus, gravity forces the grate to lift. For comparison, we present the demonstration distribution from all grasping locations in blue.

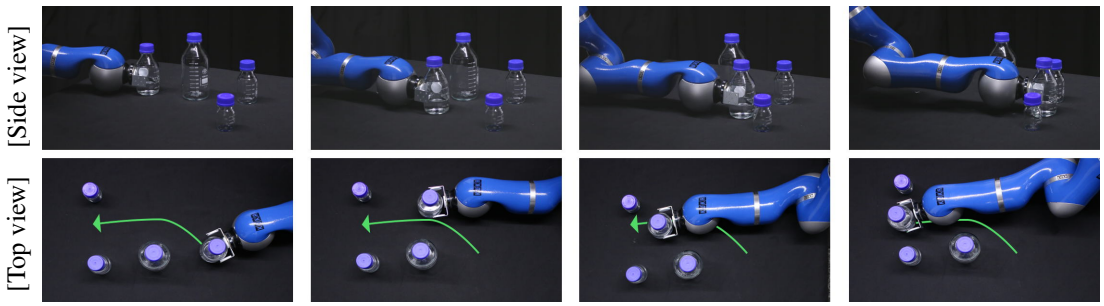


Figure 11.: Illustration of the experimental setup. The robot repositions a chemistry flask with an unknown amount of liquid, while avoiding obstacles.

4.6. *Repositioning a chemistry flask*

In this experiment we evaluate our approach on repositioning a chemistry flask filled with an unknown amount of liquid. The flask can not be moved in a straight line to the end position as another flask blocks the path to the target. Rather, the robot has to follow a curved trajectory to the end point. We used the KUKA LWR robot for the experiment. The dimensionality of state-action distribution is seventeen, seven for the degrees of freedom of the robot, three for the Cartesian forces, and seven for the controller actions. We presented the robot with two sets of ten demonstrations for two different amounts of liquid, 200ml and 400ml. Using the demonstrations, we trained a primitive that encodes the shape of the movement, the corresponding force data, and the observed actions. The interaction between the flask, the robot's end-effector, and the table cloth is not explicitly modelled.

Controlling the robot with the proposed approach, the robot reproduced the learned trajectory distribution for both liquid levels, 200ml and 400ml, without observing the amount of the liquid, as we present in 13a. Additionally, the robot reproduced successfully the task with the flask filled at $300ml$. The robot reproduces the trajectory distribution of the demonstrations. The interaction forces during the execution of the skill are shown in Fig. 14. The robot reproduces the similar interaction forces as the demonstrations. In the generalized case of filling the flask at 300ml, the interaction forces are in between the two extremes. To demonstrate the adaptation capabilities of our approach, we used conditioning to move the flask at a novel position at the end of the movement, as shown in Fig. 12, for all three liquid levels, 200ml, 300ml, and 400ml. The robot successfully reproduced the task, where the end-effector trajectories are shown in Fig. 13b. Additionally, we compare the our approach with Long Short Term Memory (LSTM) networks and Gaussian Processes (GP). In both LSTM and GP, the learning algorithm produces the torque, given the state of the system. We refer to the results of [24], where they compare both approaches using the dataset from this experiment. While both LSTM and GPs can reproduce the task, our approach enables to generalize the movement to new situations. Our approach achieves lower mean square error on average due to the lower number of model parameters. This is shown in Figure 14. However, in [24] the main motivation for using LSTMs was the ability to train dynamics models from large datasets. Whereas in this work we focused on learning motor skills from few demonstrations.

5. Conclusion

In this paper, we presented a model-free approach for Probabilistic Movement Primitives (ProMP) that can be used for learning skills for physical interaction with the environment from demonstrations. Our approach neither requires a known model of the system dynamics nor attempts to explicitly train one. Rather, we correlate the actions present during the demonstrations to the state of the robot. We showed how our approach could adapt to changes in the environment, as for example adding via-points, or placing the object to different initial positions. We showed that the model-free ProMP approach inherits many beneficial properties of MPs such as reproducing the variability in the demonstrations as well as using probabilistic

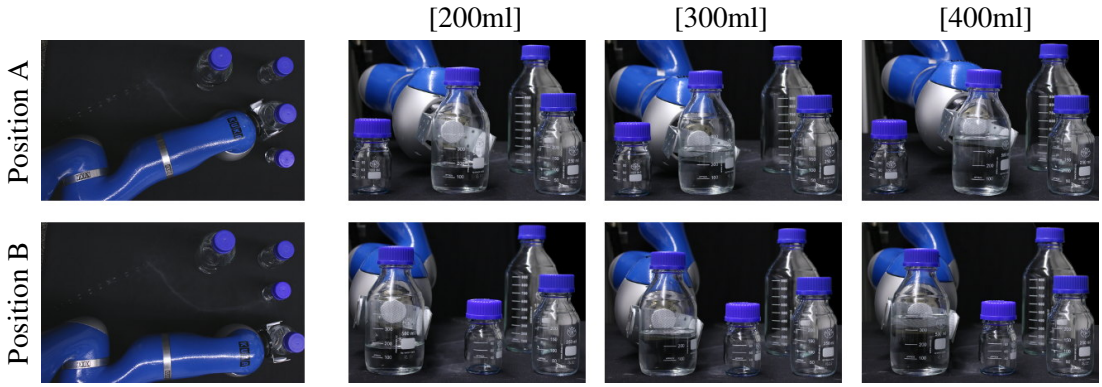
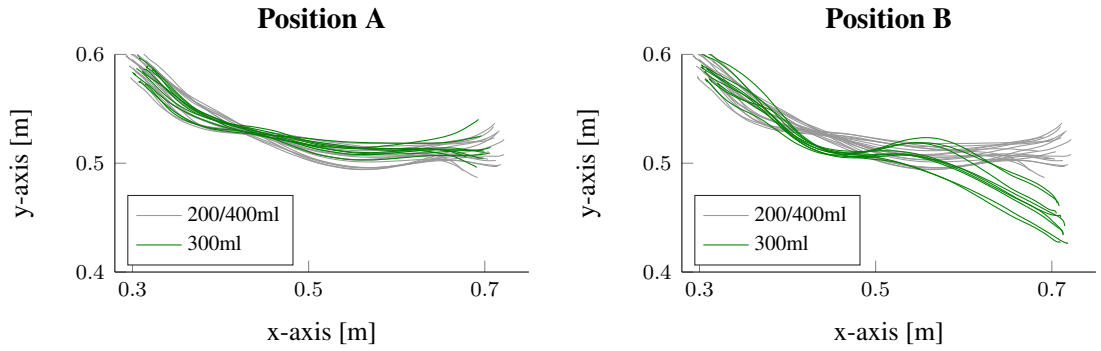


Figure 12.: Reproduction and generalization of the repositioning skill. We present the configuration of the robot at the end of the movement. The robot was trained for "Position A" and with 200ml and 400ml in the flask. Using our approach, the robot can succesfuly reproduce the movements and generalize to a different liquid amount, 300ml, and to a different final location, "Position B".



(a) Imitation position (b) Generalization position

Figure 13.: The trajectories of the end-effector of the robot at the flask repositioning experiment. We present the training data for both liquid amount, 200ml and 400ml in gray. The reproduction for the 300ml is shown in green. In (a), the robot reproduced the movement achieving the same end-effector distribution as in the demonstrations. In (b), we adapted the final location with conditioning.

operations such as conditioning for generalization to different via points. Our approach is different from directly encoding the actions, as generates the action through a model that depends on the state and the time. Hence, our approach can generalize well in the vicinity of the demonstrations. We derived a stochastic feedback controller that is obtained from the distribution over the trajectories and accurately tracks the demonstrated distribution. Our approach is best suited in tasks where time is critical for the execution of the task, e.g. pushing a button at a specific movement, or grasping a moving object. Additionally, our approach learns implicitly a local physical model nearby the demonstrations and, therefore, can be trained efficiently from a few demonstrations. However, a drawback of learning local model is that if the system is in an area away from the demonstrations, e.g. due to strong perturbations, it might be difficult for our approach to drive the system back to the learned area.

For learning physical interaction tasks, we showed that we can include sensory signals, for example the measure torques, in our distribution. By learning the correlations of this sensory signal, we can coordinate the controls needed for the physical interaction with the measured torques and forces. Such coordination is essential for the complex interaction tasks.

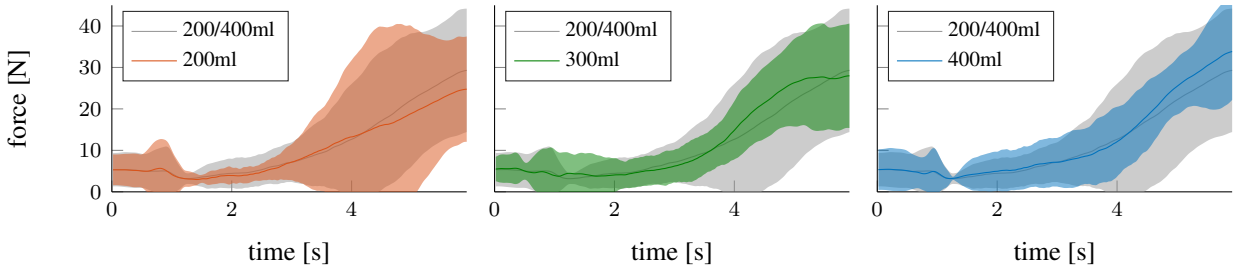


Figure 14.: We present the force distribution during the execution of the flask repositioning skill. The force profiles exerted during the demonstrations for both liquid amount, 200ml and 400ml, are shown in gray. The reproduction force profiles for both training liquid amounts and the new 300ml level, are within the demonstration distribution.

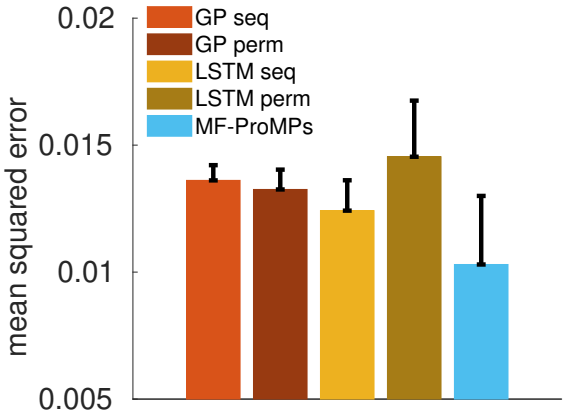


Figure 15.: We evaluate our approach on producing the desired torque vector for controlling the robot when generalizing and compared it to GPs and LSTM. We extend the comparison originally presented in [24]. The plot shows the normalized mean square error during the generalization to the 300ml case and evaluates GPs and LSTM using sequential or randomly permuted data.

Acknowledgements

The research leading to these results has received funding from the European Community's Seventh Framework Programme (FP7/2007–2013) under grant agreements #600716 (CoDyCo) and #270327 (CompLACS)

References

[1] Kosuge K, Hayashi T, Hirata Y, et al. Dance partner robot - ms dancer. In: Int. Conf. on Intelligent Robots and Systems (IROS); Oct; Vol. 4; 2003. p. 3459–3464.
[2] Maeda Y, Takahashi A, Hara T, et al. Human-robot cooperative rope turning–an example of mechanical coordination through rhythm entrainment. Advanced Robotics. 2003;17:67–78.
[3] Kim CH, Yonekura K, Tsujino H, et al. Physical control of the rotation of a flexible object – rope turning with a humanoid robot. Advanced Robotics. 2011;25:491–506.
[4] Ijspeert AJ, Nakanishi J, Schaal S. Learning attractor landscapes for learning motor primitives. In: Advances in neural information processing systems (NIPS). MIT Press; 2003. p. 1547–1554.
[5] Schaal S, Peters J, Nakanishi J, et al. Learning movement primitives. In: Int. symp. on robotics research.; 2005. p. 561–572.
[6] Billard A, Calinon S, Dillmann R, et al. Robot programming by demonstration. In: Springer handbook of robotics. Springer; 2008. p. 1371–1394.
[7] Kober J, Peters J. Learning motor primitives for robotics. In: Int. Conf. on Robotics and Automation (ICRA); 2009.

[8] Ijspeert AJ, Nakanishi J, Hoffmann H, et al. Dynamical movement primitives: learning attractor models for motor behaviors. Neural Computation. 2013;:328–373.
[9] Pastor P, Righetti L, Kalakrishnan M, et al. Online movement adaptation based on previous sensor experiences. In: Int. Conf. on Intelligent Robots and Systems (IROS); 2011. p. 365–371.
[10] Kalakrishnan M, Righetti L, Pastor P, et al. Learning force control policies for compliant manipulation. In: Int. Conf. on Intelligent Robots and Systems (IROS); 2011. p. 4639–4644.
[11] Gams A, Nemec B, Ijspeert AJ, et al. Coupling movement primitives: Interaction with the environment and bimanual tasks. IEEE Transactions on Robotics. 2014;:816–830.
[12] Paraschos A, Daniel C, Peters J, et al. Probabilistic movement primitives. In: Burges CJC, Bottou L, Welling M, et al., editors. Advances in neural information processing systems (NIPS). Curran Associates, Inc.; 2013. p. 2616–2624.
[13] Lee AX, Lu H, Gupta A, et al. Learning force-based manipulation of deformable objects from multiple demonstrations. In: Int. conf. on robotics and automation (ICRA); 2015.
[14] Paraschos A, Rueckert E, Peters J, et al. Model-free probabilistic movement primitives for physical interaction. In: Int. Conf. on Intelligent Robots and Systems (IROS); 2015.
[15] Paraschos A, Neumann G, Peters J. A probabilistic approach to robot trajectory generation. In: "Proceedings of the International Conference on Humanoid Robots (HUMANOIDS); 2013.
[16] Gams A, Do M, Ude A, et al. On-line periodic movement and force-profile learning for adaptation to new surfaces. In: Int. Conf. on Humanoid Robots (Humanoids); 2010. p. 560–565.
[17] Kormushev P, Nenchev DN, Calinon S, et al. Upper-body kinesthetic teaching of a free-standing humanoid robot. In: Int. Conf. on Robotics and Automation (ICRA); 2011. p. 3970–3975.
[18] Calinon S, D'halluin F, Sauser EL, et al. Learning and reproduction of gestures by imitation. IEEE Robotics and Automation Magazine. 2010 Jun;:44–54.
[19] Kormushev P, Calinon S, Caldwell DG. Imitation learning of positional and force skills demonstrated via kinesthetic teaching and haptic input. Advanced Robotics. 2011;:581–603.
[20] Kronander K, Billard A. Learning compliant manipulation through kinesthetic and tactile human-robot interaction. IEEE Transactions on Haptics. 2013;.
[21] Calinon S, Bruno D, Caldwell DG. A task-parameterized probabilistic model with minimal intervention control. In: Int. Conf. on Robotics and Automation (ICRA); 2014. p. 3339–3344.
[22] Evrard P, Gribovskaya E, Calinon S, et al. Teaching physical collaborative tasks: object-lifting case study with a humanoid. In: Int. Conf. on Humanoid Robots (Humanoids); 2009. p. 399–404.
[23] Gribovskaya E, Kheddar A, Billard A. Motion learning and adaptive impedance for robot control during physical interaction with humans. In: Int. Conf. on Robotics and Automation (ICRA); 2011. p. 4326–4332.
[24] Rueckert E, Nakatenus M, Tosatto S, et al. Learning inverse dynamics models in o(n) time with lstm networks. In: Proceedings of the International Conference on Humanoid Robots (HUMANOIDS); 2017.