2022 31st IEEE International Conference on Robot and
Human Interactive Communication (RO-MAN)
August 29 - September 2, 2022. Naples, Italy.



# Continuous and Incremental Learning in physical Human-Robot Cooperation using Probabilistic Movement Primitives

Daniel Schäle[1], Martin F. Stoelen[1,2] and Erik Kyrkjebø[1]

***Abstract*— For a successful deployment of physical Human-Robot Cooperation (pHRC), humans need to be able to teach robots new motor skills quickly. Probabilistic movement primitives (ProMPs) are a promising method to encode a robot's motor skills learned from human demonstrations in pHRC settings. However, most algorithms to learn ProMPs from human demonstrations operate in batch mode, which is not ideal in pHRC when we want humans and robots to work together from even the first demonstration. In this paper, we propose a new learning algorithm to learn ProMPs incrementally and continuously in pHRC settings. Our algorithm incorporates new demonstrations sequentially as they arrive, allowing humans to observe the robot's learning progress and incrementally shape the robot's motor skill. A built-in forgetting factor allows for corrective demonstrations resulting from the human's learning curve or changes in task constraints. We compare the performance of our algorithm to existing batch ProMP algorithms on reference data generated from a pick-and-place task at our lab. Furthermore, we demonstrate how the forgetting factor allows us to adapt to changes in the task. The incremental learning algorithm presented in this paper has the potential to lead to a more intuitive learning progress and to establish a successful cooperation between human and robot faster than training in batch mode.**

## I. Introduction

Physical Human-Robot Cooperation (pHRC) has great potential for the (semi) automation of manufacturing processes with small batch sizes and frequently changing tasks. Due to the cognitive abilities of the human in the loop, pHRC is expected to be more versatile and flexible than conventional (full) automation and could offer suitable automation concepts to small and medium sized enterprises. The pHRC tasks we consider in this paper can only be completed with frequent or continuous physical interactions between human and robot. The control throughout the task is shared between both partners and both should contribute equally to the work progress. The intention of pHRC is that the human and robot work and learn together on shared tasks, hence learning frameworks for pHRC should not be characterized by minimizing the training time until the robot becomes autonomous in its operation, but rather by the time until a successful cooperation between human and robot evolves. Thus, cooperative learning in pHRC differs from other types of robot learning of motor skills where the goal is autonomous robot operation. In pHRC, the focus lays on continuous learning of motor skills, where the human and robot can start working together even from the first demonstration, and where the robot continuously learns and improves motor skills from cooperating with a human.

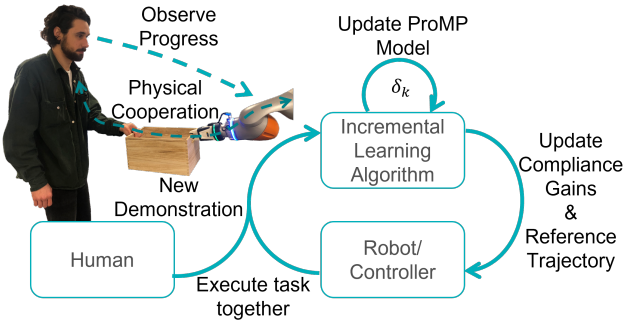


Fig. 1. The intended use for our incremental learning algorithm for ProMPs in physical Human-Robot Cooperation. Human and robot learn a new task together by trying to solve the task. Each task execution serves as a new demonstration which is incrementally incorporated into the robot's motor skill by our algorithm. The human can observe the robot's learning progress and thus incrementally shape the robot's motor skill.

A common approach within the robotics community to teach motor skills to a robot is Learning from Demonstration (LfD) in combination with movement primitives [1]. However, in pHRC, learning new tasks not only means the acquisition of motor skills, but also to learn interaction controllers for successful cooperation. An important aspect is assigning leader-follower roles appropriate to the skills of human and robot, where the robot should take control and be stiff in sections that require precision or physical strength, whereas the human should take control over a passive/soft robot when cognitive skills and decision making are required. As shown in e.g. [2], this cooperative intelligence can be implemented in terms of impedance control with gains computed from the variance in the task executions.

A promising movement primitive framework for pHRC are Probabilistic Movement Primitives (ProMPs) [3]. First, their probabilistic representation of motion captures the variance in the executions of a task, which in turn gives hints about the required precision and accordingly, the desired stiffness of the robot. Second, the ProMP framework provides several operators to condition, combine and sequence motor skills, which makes it easier to generalize ProMPs to unknown situations which occur frequently when robots interact with humans in unstructured environments. However, most of the algorithms in the literature to train ProMPs from human demonstrations operate in batch mode [4]–[6], which we argue is suboptimal for pHRC settings. Batch mode means that all human demonstrations of a new task are provided at the beginning of the training. Once the desired number (batch

Funded by the Research Council of Norway through grant number 28077.
[1]Department of Computer Science, Electrical Engineering and Mathematical Sciences, Faculty of Engineering and Science, Western Norway University of Applied Sciences, Førde, Norway. dasc@hvl.no
[2]CRNS, University of Plymouth, Plymouth, United Kingdom.

size) of demonstrations is reached, the learning algorithm calculates the ProMP parameters, and the human and the robot can start to work on the actual task. In batch mode, the training phase and execution phase of the task are separated, which delays the overall progress of the task. Also, most humans will only have a vague understanding of how a robot learns new motor skills and how it reacts to certain demonstrations. Thus, it may be difficult for lay users to provide a set of demonstrations which models the robot's desired behavior without observing any intermediate training progress. Updating a motor skill with new demonstrations becomes increasingly difficult in batch mode since computation time and memory requirements increase with the number of demonstrations. Hence, we claim that batch learning can not readily be used for lifelong learning with continuous shaping of motor skills.

Batch algorithms based on the expectation-maximization algorithm (EM) have been proposed by [4] for maximum likelihood estimates (MLE) of single ProMPs and [5] for maximum a posteriori estimates (MAP) of single ProMPs. The authors in [4] and [5] treat weight vectors as hidden variables and use EM to maximize the marginal likelihood of the observed robot states with respect to the distribution parameters of the ProMP. The main difference between [4] and [5] is that [5] regularize their parameter estimates by means of a prior distribution over the ProMP parameters. The general advantage of the approach in [4] and [5] over algorithms that compute the weights by least squares (such as [6]–[8]) is the increased robustness against noisy and incomplete demonstration data [4], [5]. Apart from that, the prior parameter distribution in [5] offers a convenient way of regularization and to incorporate prior assumptions about the ProMP parameters. In [9], prior parameter distributions are used to enrich ProMPs with contextual information inspired by speed-accuracy trade-offs found in human motion. Regularization and incorporating prior knowledge are both important for incremental learning, due to the sparse data at the beginning of training.

An incremental learning algorithm for a Gaussian mixture model of Interaction ProMPs (IProMPs) was proposed by [7]. IProMPs are a variant of ProMPs which are learned from demonstrations containing a robot trajectory and a corresponding human trajectory recorded via a motion capturing system. The resulting joint model, the IProMP, is used to make the robot respond appropriately to new human motion inputs in an interaction scenario. In case of a new demonstration, [7] compute a point estimate of the weight vector via least squares and then update the Gaussian mixture model with an EM algorithm based on [10]. In [11], an incremental learning algorithm with a forgetting factor is presented. However, the movement primitives in [11] are different from ProMPs and based on Hidden Markov Models.

In this paper, we want to overcome the aforementioned limitations of batch learning with an incremental learning algorithm for ProMPs which allows human and robot to jointly learn new motor skills in pHRC settings. The intended use of our incremental learning algorithm is in combination with a feedback controller with time-varying stiffness gains. The controller executes the learned ProMP on the robot after the first and each following demonstration, see Fig. 1. This means that the robot starts contributing to the task already after the first demonstration and the human can focus more on the task and correcting the robot when needed instead of providing full demonstrations. Each execution of the task naturally becomes the next demonstration. The stiffness gains are computed based on the variance of the ProMP, such that the robot will be soft/passive at the beginning and increasingly take over control and be stiff in sections of the task that require precision. In this process, the human-robot interaction can transition smoothly from pure LfD into a physical cooperation. The driving factors behind our algorithm are to enable a successful cooperation from the first demonstration onwards, and to create an intuitive and open-ended learning progress. Our algorithm incorporates new demonstrations sequentially as they arrive, allowing the human to incrementally shape the robot's motor skill. Thus, the user can observe how the robot reacts and adapts its behavior to each demonstration and how well the robot performs already with respect to the task objective. In tasks new to both human and robot, the human will be subject to a learning curve as well. We allow for improvements of the human's demonstrations by introducing a forgetting factor into our algorithm [12], which discounts the influence of older demonstrations to the ProMP. Furthermore, the forgetting factor makes it possible to adapt a motor skill to changes in the task constraints in later stages of the cooperation. The forgetting factor is essential for the robot to regain confidence and control after e.g. the change of a via-point position, since the variance will not readily converge to this new situation without forgetting demonstrations from before the change. With the aim to learn interaction controllers from the variance, it is not desired to learn a ProMP which encompasses all previous demonstrated variations of a task, but rather to keep track of a ProMP that represents the current status of the cooperation well. The objective for the work is not to teach the robot to become autonomous in it's operations through the demonstrations, but for the robot to quickly learn how to cooperate with the human in pHRC to solve a task together – and where it is the cooperation between the human and robot that must be learnt.

For our purpose, none of the training algorithms found in the literature are optimal. We want to profit from the robustness of the two batch algorithms, while on the other hand, pHRC settings require an incremental and open-ended training progress. Furthermore, we do not want to use IProMPs, since we want to avoid using a motion capturing system, and instead explore a cooperative learning process based only on the physical interaction between human and robot. Note that in this paper, we focus on learning single ProMPs instead of libraries in terms of mixture models.

## II. Incremental Learning

In this section we introduce our algorithm for the incremental learning of ProMPs in pHRC settings: First, we

review relevant aspects of the ProMP framework [3], and second, we describe how we use an online variant of an expectation maximization algorithm to learn ProMP parameters from human demonstration.

### A. Probabilistic Movement Primitives

A ProMP represents a distribution over trajectories [3]. A trajectory $\boldsymbol{\tau} = \{\boldsymbol{y}_t\}_{t=1}^{T}$ is a time-series of vector-valued robot states $\boldsymbol{y}_t \in \mathcal{S}$ in a state space $\mathcal{S} \subseteq \mathbb{R}^D$, where $D$ is the dimension of the state space. Both joint space and task space are valid choices for $\mathcal{S}$. To reduce the number of model parameters, trajectories are concisely represented as weight vectors in a basis function model

$$\boldsymbol{y}_t = \boldsymbol{\Phi}_t \boldsymbol{w} + \boldsymbol{\epsilon}_y \ . \tag{1}$$

A weight vector $\boldsymbol{w} \in \mathbb{R}^{KD}$ is related at time $t$ to the robot's state $\boldsymbol{y}_t$ through a time dependent, block diagonal basis function matrix $\boldsymbol{\Phi}_t \in \mathbb{R}^{D \times KD}$. The matrix $\boldsymbol{\Phi}_t$ contains on its diagonal a row vector $\boldsymbol{\phi}_{d,t}^{\intercal} \in \mathbb{R}^K$ for each degree of freedom, which again contains the values of $K$ normalized, evenly spaced, Gaussian basis functions $\phi_k(t)$ evaluated at time $t$. The weight vector $\boldsymbol{w}$ is a vertical concatenation of $D$ column vectors $\boldsymbol{w}_d \in \mathbb{R}^K$, representing the weight vectors of each individual degree of freedom of the robot. The last term $\boldsymbol{\epsilon}_y \in \mathbb{R}^D$ is a vector containing the observation noise which is assumed to be independent and identically distributed and to follow the normal distribution $\mathcal{N}(\boldsymbol{0}, \boldsymbol{\Sigma}_y)$.
Given a weight vector $\boldsymbol{w}$, it follows that a trajectory $\boldsymbol{\tau}$ consisting of $T$ time steps is distributed according to

$$p(\boldsymbol{\tau}|\boldsymbol{w}) = \textstyle\prod_{t=1}^{T} \mathcal{N}(\boldsymbol{y}_t|\boldsymbol{\Phi}_t \boldsymbol{w}, \boldsymbol{\Sigma}_y) \ . \tag{2}$$

Multiple demonstrations of the same movement are expected to differ slightly. This implies that different weight vectors $\boldsymbol{w}_n$ are needed to represent the $n$ different instances of a movement. The underlying mechanism generating the weight vector samples is assumed to be a Gaussian distribution

$$p(\boldsymbol{w}|\boldsymbol{\theta}_w) = \mathcal{N}(\boldsymbol{w}|\boldsymbol{\mu}_w, \boldsymbol{\Sigma}_w) \ , \tag{3}$$

where $\boldsymbol{\theta}_w = \{\boldsymbol{\mu}_w, \boldsymbol{\Sigma}_w\}$ are the distribution parameters. The mean vector $\boldsymbol{\mu}_w \in \mathbb{R}^{KD}$ summarizes the mean of the demonstrations in each degree of freedom. The covariance matrix $\boldsymbol{\Sigma}_w \in \mathbb{R}^{KD \times KD}$ represents the variances and covariances of the demonstrations in respectively between each degree of freedom. Learning a ProMP from demonstration requires to find the distribution parameters $\boldsymbol{\theta}_w$ that explain the demonstration data best.

The complete ProMP model is obtained by combining Eq. 2 and 3 and subsequently marginalizing out the weights $\boldsymbol{w}$. A movement encoded as a ProMP with parameters $\boldsymbol{\theta}_w = \{\boldsymbol{\mu}_w, \boldsymbol{\Sigma}_w\}$, is represented by the distribution

$$p(\boldsymbol{\tau}|\boldsymbol{\theta}_w) = \textstyle\int \mathcal{N}(\boldsymbol{w}|\boldsymbol{\mu}_w, \boldsymbol{\Sigma}_w) \prod_{t=1}^{T} \mathcal{N}(\boldsymbol{y}_t|\boldsymbol{\Phi}_t \boldsymbol{w}, \boldsymbol{\Sigma}_y) d\boldsymbol{w} \ . \tag{4}$$

The time signals of the demonstrations are normalized to compensate for different durations. For this purpose, the time signal $t$ is replaced by a phase variable $z_t$ that ensures that every demonstration runs between $z_0 = 0$ and $z_T = 1$.

### B. Incremental Learning Algorithm

In this paper, we propose to use an online variant of the EM algorithm known as stepwise EM (sEM) [12], [13] to train ProMPs incrementally in pHRC settings. As usual in EM, the objective of the algorithm is to maximize the likelihood of the observed data $\boldsymbol{Y} = \{\boldsymbol{\tau}_n\}_{n=1}^{N}$, consisting of $N$ trajectories $\boldsymbol{\tau}_n$, with respect to the model parameters $\boldsymbol{\theta}_w$. The weight vectors $\boldsymbol{w}_n$ are treated as hidden variables. The marginal likelihood is given by

$$p(\boldsymbol{Y}|\boldsymbol{\theta}_w) = \textstyle\prod_{n=1}^{N} \int p(\boldsymbol{w}_n|\boldsymbol{\mu}_w, \boldsymbol{\Sigma}_w) \prod_{t=1}^{T} p(\boldsymbol{y}_{nt}|\boldsymbol{w}_n) d\boldsymbol{w}_n \ . \tag{5}$$

For an effective use of the sEM algorithm we exploit the properties of exponential family distributions. The joint distribution of observed and hidden variables $p(\boldsymbol{\tau}, \boldsymbol{w}|\boldsymbol{\theta}_w)$ for a single trajectory $\boldsymbol{\tau}$ is an exponential family with sufficient statistics $\boldsymbol{s}_1 = \boldsymbol{w}$, $\boldsymbol{s}_2 = \boldsymbol{w}\boldsymbol{w}^{\intercal}$ and $\boldsymbol{s}_3 = \sum_{t=1}^{T} \boldsymbol{y}_t \boldsymbol{y}_t^{\intercal} - 2\boldsymbol{y}_t \boldsymbol{w}^{\intercal} \boldsymbol{\Phi}_t^{\intercal} + \boldsymbol{\Phi}_t \boldsymbol{w}\boldsymbol{w}^{\intercal} \boldsymbol{\Phi}_t^{\intercal}$. The sufficient statistics $\boldsymbol{s}$ offer a convenient way to summarize arbitrary amounts of demonstrations without loss of information. This is a useful property for online learning in terms of EM, since we can compute the expected sufficient statistics (ESS) in the E-step, accumulate them in some form as new demonstrations arrive, and then compute the MLE or MAP of the ProMP parameters from the accumulated ESS in the M-step.

The pseudo code for the proposed algorithm is shown in Algorithm 1. The ESS $\boldsymbol{u}'$ are computed as the expected values of the sufficient statistics $\boldsymbol{s}$ based on the posterior distribution over the hidden variables $p(\boldsymbol{w}|\boldsymbol{\tau}, \boldsymbol{\theta}_w^{old})$ given the current model parameters $\boldsymbol{\theta}_w^{old}$. The posterior distribution over the hidden variables is computed in line 5 and 6; the ESS in line 7 to 9. In sEM, whenever a new demonstration is added, the ESS are computed solely for the new demonstration. Previously added demonstrations are not stored and visited again. For comparison: in each iteration of batch EM the computations in the E-step are performed for the entire data set. Since the ESS for a single demonstration would be a poor approximation to the ESS across the complete data set, the sEM algorithm interpolates between the sum of all previous statistics $\boldsymbol{u}$ and the statistics of the latest demonstration $\boldsymbol{u}'$. This interpolation is done in line 10 in Algorithm 1. The resulting weighted sum of old and new statistics is then used to update the model parameters $\boldsymbol{\theta}_w$ in the M-step. For the MAP estimation in the M-Step in line 14 and 16 we use a normal-inverse-Wishart prior, the conjugate prior of the multivariate Gaussian distribution $p(\boldsymbol{w}|\boldsymbol{\theta}_w)$. See [5], [9] for a detailed description.

The interpolation in line 10 is governed by a step size $\delta_N$, which controls the strength of the sum of the previous statistics $\boldsymbol{u}$ over the new statistics $\boldsymbol{u}'$. Similar to [13], we use a step size $\delta_N = (N+1)^{-\beta}$ where $0.5 < \beta \leq 1$, which decays with the number of demonstrations or parameter updates $N$. The user-defined parameter $\beta$ is used to tune the decay. A smaller $\beta$ will lead to larger, slowly decaying step sizes $\delta_N$ and hence, to larger updates to the statistics $\boldsymbol{u}$.

Note that all computations can be performed without storing the previous demonstrations and their ESS, which

**Algorithm 1:** Stepwise EM Algorithm for Training ProMPs incrementally in pHRC settings.

**Data:** A new demonstration $\boldsymbol{\tau} = \{\boldsymbol{y}_t, z_t\}_{t=1}^{T'}$ containing the robot states $\boldsymbol{y}_t$ and corresponding normalized time stamps $z_t$.

**Input:** Step size reduction power $0.5 < \beta \leq 1$, initial values for $\boldsymbol{\mu}_w, \boldsymbol{\Sigma}_w, \boldsymbol{\Sigma}_y$. Typically $\boldsymbol{\mu}_w = \mathbf{0}$, $\boldsymbol{\Sigma}_w = \boldsymbol{I}$, $\boldsymbol{\Sigma}_y = \boldsymbol{I}$.

**Output:** ProMP parameters $\boldsymbol{\mu}_w, \boldsymbol{\Sigma}_w, \boldsymbol{\Sigma}_y$.

Initialize $\boldsymbol{u} = \mathbf{0}$, $\eta = 0$, $T = 0$, $N = 1$
Compute initial step size $\delta_N \leftarrow (N+1)^{-\beta}$
**if** *new data* $\boldsymbol{\tau}$ *available* **then**
  $\boldsymbol{\Phi}_t \leftarrow \boldsymbol{\Phi}(z_t)\ \forall t$
  $\boldsymbol{S}_w \leftarrow \left(\boldsymbol{\Sigma}_w^{-1} + \sum_{t=1}^{T} \boldsymbol{\Phi}_t^\intercal \boldsymbol{\Sigma}_y^{-1} \boldsymbol{\Phi}_t\right)^{-1}$
  $\overline{\boldsymbol{w}} \leftarrow \boldsymbol{S}_w \left(\boldsymbol{\Sigma}_w^{-1} \boldsymbol{\mu}_w + \sum_{t=1}^{T} \boldsymbol{\Phi}_t^\intercal \boldsymbol{\Sigma}_y^{-1} \boldsymbol{y}_t\right)$
  $\boldsymbol{u}_1' \leftarrow \overline{\boldsymbol{w}}$
  $\boldsymbol{u}_2' \leftarrow \overline{\boldsymbol{w}}\overline{\boldsymbol{w}}^\intercal + \boldsymbol{S}_w$
  $\boldsymbol{u}_3' \leftarrow \sum_{t=1}^{T'} \boldsymbol{y}_t \boldsymbol{y}_t^\intercal - 2\boldsymbol{y}_t \overline{\boldsymbol{w}}^\intercal \boldsymbol{\Phi}_t^\intercal + \boldsymbol{\Phi}_t (\overline{\boldsymbol{w}}\overline{\boldsymbol{w}}^\intercal + \boldsymbol{S}_w) \boldsymbol{\Phi}_t^\intercal$
  $\boldsymbol{u} \leftarrow (1 - \delta_N)\boldsymbol{u} + \delta_N \boldsymbol{u}'$
  $\eta \leftarrow (1 - \delta_N)\eta + \delta_N$
  $T \leftarrow (1 - \delta_N)T + \delta_N T'$
  $\boldsymbol{\mu}_w^* \leftarrow \frac{1}{\eta}\boldsymbol{u}_1$
  $\boldsymbol{\mu}_w \leftarrow \frac{1}{N+k_0}(k_0 m_0 + N\boldsymbol{\mu}_w^*)$
  $\boldsymbol{\Sigma}_w^* \leftarrow \frac{1}{\eta}\boldsymbol{u}_2 - \boldsymbol{\mu}_w \boldsymbol{\mu}_w^\intercal$
  $\boldsymbol{\Sigma}_w \leftarrow \frac{\boldsymbol{S}_0 + N\boldsymbol{\Sigma}_w^* + \frac{k_0 N}{k_0 + N}(\boldsymbol{\mu}_w^* - \boldsymbol{m}_0)(\boldsymbol{\mu}_w^* - \boldsymbol{m}_0)^\intercal}{N + v_0 + KD + 2}$
  $\boldsymbol{\Sigma}_y \leftarrow \frac{1}{T}\boldsymbol{u}_3$
  $N \leftarrow N + 1$
  $\delta_N \leftarrow (N+1)^{-\beta}$
  **return** $\boldsymbol{\mu}_w, \boldsymbol{\Sigma}_w, \boldsymbol{\Sigma}_y$
**end**

means that the memory use of the algorithm is constant. The algorithm can be modified to train on mini-batches instead of single demonstrations [13]. Training in mini-batches has the advantage to produce more stable updates but slows down the incremental training progress by pooling a few demonstrations before updating the ProMP parameters. We have omitted the mini-batches here for brevity.

The step size $\delta_N$ serves as a forgetting factor in our learning algorithm. More precisely, the step size weighs the contribution of ESS computed several iterations ago less. This is important in two regards for the incremental learning setting under our consideration: First, from a probabilistic perspective, it has to be kept in mind that in each iteration the ESS are computed using the posterior distribution over the hidden variables under the *current* ProMP parameters. It follows that the posterior distributions, and hence the ESS, computed in the first few iterations are potentially far away from their true value, since they were computed based on crude estimates of the model parameters. It is crucial to forget those crude early estimates during the training, such that the ESS computed after a number of updates to the model parameters quickly dominate the ESS from early iterations and the estimated model parameters approach their true values faster. Second, from a motor learning perspective, discounting the contributions from early demonstrations to a motor skill is important. As described earlier, the quality of the human demonstration is expected to increase as the learning of a new motor skill proceeds. Apart from that, forgetting demonstrations which where provided a long time ago helps to adapt a motor skill to gradual changes in the task structure. Note that for an open-ended shaping of a motor skill, the step size should approach a value greater than zero.

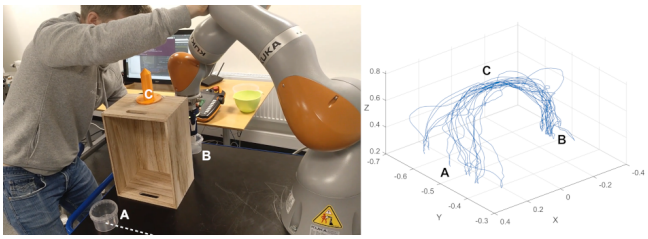


Fig. 2. Pick and place task used to generate a reference ProMP. The task objective was to move the robot's end effector between cup A and B, while passing the orange tip C precisely with the gripper fingers. To introduce variance in the task constraints, Cup A was moved along the dotted line during the trials. **Left:** Task setting with KUKA iiwa 14 manipulator. **Right:** Recorded end-effector trajectories in task space.

## III. Experimental Evaluation

For the experimental evaluation of our algorithm we generated a reference ProMP based on data of a pick-and-place task done with a KUKA LBR iiwa 14 R820 robot. The task setting and recorded end-effector trajectories in task space are shown in Fig. 2. The reference ProMP has $D = 3$ DOF and uses $K = 10$ basis functions, yielding a mean vector $\boldsymbol{\mu}_w^{ref} \in \mathbb{R}^{30}$, a covariance matrix $\boldsymbol{\Sigma}_w^{ref} \in \mathbb{R}^{30\times30}$ and a observation noise covariance matrix $\boldsymbol{\Sigma}_y^{ref} \in \mathbb{R}^{3\times3}$. We generated the reference parameters based on the empirical distribution of the robot data downsampled to $K$ time steps. The observation noise was sampled from an inverse Wishart distribution. We sampled $N = 100$ demonstrations from this reference ProMP to generate the training data set shown in Fig. 3, and used in the following experiments. In the experiments in this paper, we compare the pure LfD performance of all algorithms, and therefore do not use the proposed algorithm in combination with a robot controller.

### A. Comparison of Training Algorithms

In the first experiment, we compare the performance of different training algorithms for ProMPs from literature with the incremental algorithm presented in this paper. The performance of the different algorithms can be assessed by comparing the estimated parameters to the reference. For a quantitative quality measure of the estimated parameters we use the Bhattacharyya distance $D_B$ to compute the distance between the distribution of the reference ProMP to the ProMP distribution of the algorithm tested. Also, we look at the matrix condition number $\kappa$ of the covariance matrix $\boldsymbol{\Sigma}_w$

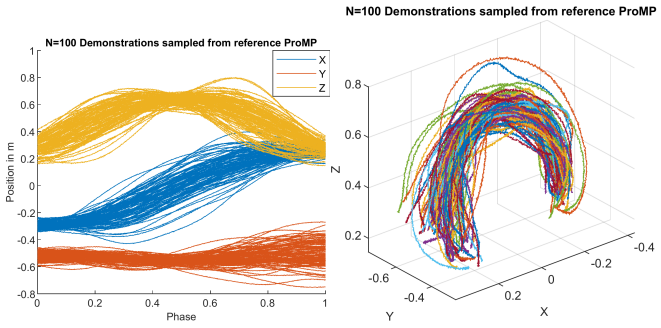


Fig. 3. The training data set used in our experiments: $N = 100$ Demonstrations sampled from a reference ProMP. **Left:** X,Y and Z components of the sampled demonstrations plotted over the movement phase $z_t$. **Right:** sampled demonstrations shown in task space.

to asses the numerical stability of the estimated covariance matrices. The parameter $\kappa$ is defined as the ratio of the largest singular value of $\boldsymbol{\Sigma}_w$ to the smallest. A large condition number indicates a nearly singular matrix, which can not be inverted reliably. Estimating full covariance matrices is generally difficult, especially with a low number of data in relation to the matrix size. However, invertible covariance matrices are crucial for a number of operations in the ProMP framework; thus, the numerical stability of $\boldsymbol{\Sigma}_w$ is a relevant performance measure for the training algorithms. The four different training algorithms MLE with Ridge Regression [6], MLE with (batch) EM [4], MAP with (batch) EM [5], and the MAP with sEM proposed in this work, were applied to the training data set. The specific parameter settings are given in Table I. For the MAP estimation in our sEM algorithm we used the same parameters for the normal-inverse-Wishart prior as the authors in [5]: $\boldsymbol{m}_0 = \mathbf{0}$, $k_0 = 0$, $v_0 = KD + 1$ and $\boldsymbol{S}_0 = (v_0 + KD + 1)\,\text{blockdiag}(\boldsymbol{\Sigma}_w^*)$.
In the last row of Table I, sEM is used as a batch algorithm, by letting it pass five times incrementally over the data. This additional test was done to compare our sEM algorithm to the EM algorithms on the condition of visiting each demonstration equally often. Note that sEM in the online setting (one pass over the data) only visits the 100 demonstrations once, while the EM-based batch algorithms perform five iterations, hence visit each demonstration five times. Using our incremental algorithm in batch mode is of course contrary to the rationale of this paper, but legitimate for a fair theoretical comparison to the batch EM algorithms.

The resulting performance measures $D_B$ and $\log\kappa(\boldsymbol{\Sigma}_w)$ after applying the training algorithms are shown in Table I. MAP estimation with EM performs best with respect to both Bhattacharyya distance $D_B$ and matrix condition number $\kappa$. Our algorithm, MAP with sEM, estimates in batch mode (5 passes) as well as online mode covariance matrices with condition numbers comparable to those of MAP with EM. In online mode, our algorithm has the greatest of all Bhattacharyya distances, though the distance is still of comparable magnitude to the other ones. However, with five passes over the data, our algorithm yields the second smallest distance and comes quite close to MAP with sEM.

TABLE I
COMPARISON OF THE TRAINING ALGORITHMS.

| | | Performance Measure | |
|---|---|---|---|
| Algorithm | Algorithm settings | $D_B$ | $\log\kappa(\boldsymbol{\Sigma}_w)$ |
| Reference | - | - | 4.1988 |
| MLE with Ridge Reg. | $\lambda = 10^{12}$ | 0.8077 | 5.6010 |
| MLE with EM | 5 Iterations | 0.8098 | 5.6046 |
| MAP with EM | 5 Iterations | 0.6787 | 4.8632 |
| MAP with sEM | $\beta = 0.75$ | 0.8978 | 4.9552 |
| MAP with sEM | $\beta = 0.75$, 5 Passes | 0.7012 | 4.9325 |

### B. Incremental Training Progress

In our second experiment, we inspect the incremental training progress of our online algorithm. It is critical to ensure that the incremental parameter estimates are reasonable and usable. We apply our algorithm in online-mode (1 pass over the data), with a step size reduction power $\beta = 0.75$ on the same training data set as before. Hence, we can once more compare the estimated ProMP parameters to those of the reference ProMP. Again, we use the same prior parameters as in [5] for MAP estimation.

We use different indicators to analyze the parameter estimates during the training. As before, we use the Bhattacharyya distance $D_B$ to measure the distance to the distribution of the reference ProMP. To analyze the mean $\boldsymbol{\mu}_w$ and covariance $\boldsymbol{\Sigma}_w$ independently, we compute the relative errors $E_F$ of the Frobenius norms of the two parameters. To assess the numerical stability of the covariance estimates, we compute the logarithm of the matrix condition number of $\boldsymbol{\Sigma}_w$. Further, we compute the rotation angle between the first principal component of the ProMP distribution before and after adding a new demonstration. Large rotations of the first principal component indicate major changes in the covariance structure, either caused by an ill-conditioned covariance matrix that is sensitive to variations in the input data, or by large actual changes in the demonstration data.

The indicators are computed each time a new demonstration is added to the ProMP. Fig. 4 shows all four indicators plotted over the course of training. The Bhattacharyya distances reach an almost steady level after around 40 demonstrations. The relative errors computed from the Frobenius norms reach near steady levels immediately for the mean $\boldsymbol{\mu}_w$ and after about 15 demonstrations for the covariance $\boldsymbol{\Sigma}_w$. To visualize the incremental training progress we show the evolution of the ProMP distribution during the training on the first 19 demonstration, as well as the final ProMP after 100 demonstrations and the reference ProMP in Fig. 5.

### C. Adaptation to Changes in the Task

In this experiment, we demonstrate how our online algorithm adapts the ProMP to shifts in the task constraints/execution. We use the same set of demonstrations as before, but manipulate its order as follows: We sort the data according to the endpoint (mean of the last five time steps) of the Y coordinate at cup A (Fig. 2). We then separate 30 demonstrations with the largest endpoint values from the remaining 70 demonstrations. The order within both

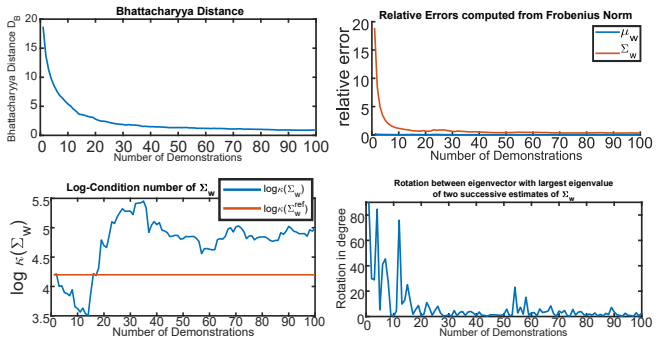


Fig. 4. Incremental training progress with sEM. **Top Left:** Bhattacharyya distance $D_B$ to distribution of reference ProMP. **Top Right:** Relative Errors of the Frobenius norm $E_F$ of the estimated ProMP parameters to the reference. **Bottom Left:** Logarithm of the matrix condition number of the estimated covariance matrix $\mathbf{\Sigma}_w$. **Bottom Right:** Rotation of the eigenvector with the highest corresponding eigenvalue (first principal component) between to two successive estimates of $\mathbf{\Sigma}_w$. Large rotations indicate major changes in the variance structure of the ProMP distribution.

subsets is randomized again. Finally, the subset containing the demonstrations with larger endpoint values is placed after the subset containing the remaining demonstrations. The resulting set of demonstrations has two distinct mean endpoints for Y; one mean for the first 70 demonstrations and one mean for the last 30 demonstrations. See Fig. 6 for clarification. This arrangement simulates a long training phase with similar task constraints in which the training algorithms converge to a set of parameters. This is then followed by a change in the task, represented by a shifted mean for the end of the movement in the Y direction. We claim that our online algorithm is well suited to capture such changes in the task execution. We compare our online algorithm with a step size reduction power $\beta = 0.6$ to the batch algorithm MAP with EM (5 iterations), which had the best performance in the comparison of the training algorithms in section III-A. Fig. 7 shows the resulting ProMPs of both algorithms together with the demonstration data. The ProMP trained with sEM is adapted to the most recent data (red lines) by means of both mean and variance. The ProMP trained with batch EM, however, represents the entire set of demonstrations, with a mean in the middle of the first (blue) and second (red) subset of demonstrations. The variance contains the entire set of demonstrations. The mean endpoints of the Y component during the course of training are shown on the right in Fig. 6. In case of sEM, the adaptation of the ProMP mean to the new endpoint is clearly visible.

## IV. Learning on Physical Robot

To show the adaptation capabilities of our algorithm on unprocessed robot data, we conducted a similar pick-and-place experiment as in section III-C on a Franka Emika Panda manipulator. The task was to pick up a part at a fixed position, move it precisely past an inspection camera and place it in a small container. To enforce a change in the task execution, the container was moved about 20cm to a new ”place”-position after 15 demonstrations, followed by a further 15 demonstrations. We applied our incremental algorithm and the batch algorithm MAP with EM on the 30 demonstrations, as described in section III-C. The resulting ProMPs are shown in Fig. 8. The mean of the incrementally trained ProMP is shifted to the new ”place”-position shown on the left of Fig. 8. In contrast, the ProMP trained with batch EM, shown on the right in Fig. 8, represents the entire distribution of demonstrations, with a mean that ends between the initial and the final ”place”-position and a variance that contains both the ”place”-positions. In this simple example, the change in the task could be handled by conditioning the ProMP on the box position, but this would however require external sensor systems. Also, the changes to a motion in pHRC and incremental learning are expected to be more involved, making the selection and detection of via-points and orientations to condition very difficult.

## V. Discussion and Conclusions

In the following section, we discuss the results from the three experiments in section III and relate them to the general context of learning motor skills incrementally in pHRC.

### A. Discussion on Training Algorithm Performance

According to the results in Table I, all algorithms lead to comparable results. As expected, the algorithms doing MAP estimation perform better in terms of the numerical stability of the covariance matrix, indicated by a lower condition number. Similar results are reported in [5]. The improved numerical stability is a consequence of the regularization through the prior parameter distribution used to compute the maximum a-posteriori estimates. MAP estimation with EM performs best of all algorithms in terms of both Bhattacharyya distance and condition number of the covariance matrix. Our incremental algorithm (MAP with sEM) has the largest Bhattacharyya distance to the reference. However, if we let it pass the data five times, as the batch EM algorithms, the incremental algorithm performs second best. The online learning progress comes at the expense of a slight loss in accuracy of the final parameter estimates. Given that the Bhattacharyya distance of our online algorithm is in the same range as the distances of the other algorithms, and if we compare the plot of the final ProMP distribution trained with the online algorithm to the reference ProMP in Fig. 5, this loss in accuracy seems to be acceptable – especially in the pHRC settings we aim for, where an incremental training progress is essential.

### B. Discussion on Incremental Training Progress

Considering the evolution of the ProMP distribution in Fig. 5, the variance structure of the first few ProMPs contains few of the characteristics of the final ProMP and is not descriptive of the task. E.g., the via-point in the middle of the movement that has to be passed with high accuracy is not visible. After around 15 demonstrations, the ProMP shows good correspondence to the final ProMP trained with all 100 demonstrations. This correspondence after 15 demonstrations is also reflected in the relative error of the Frobenius norm

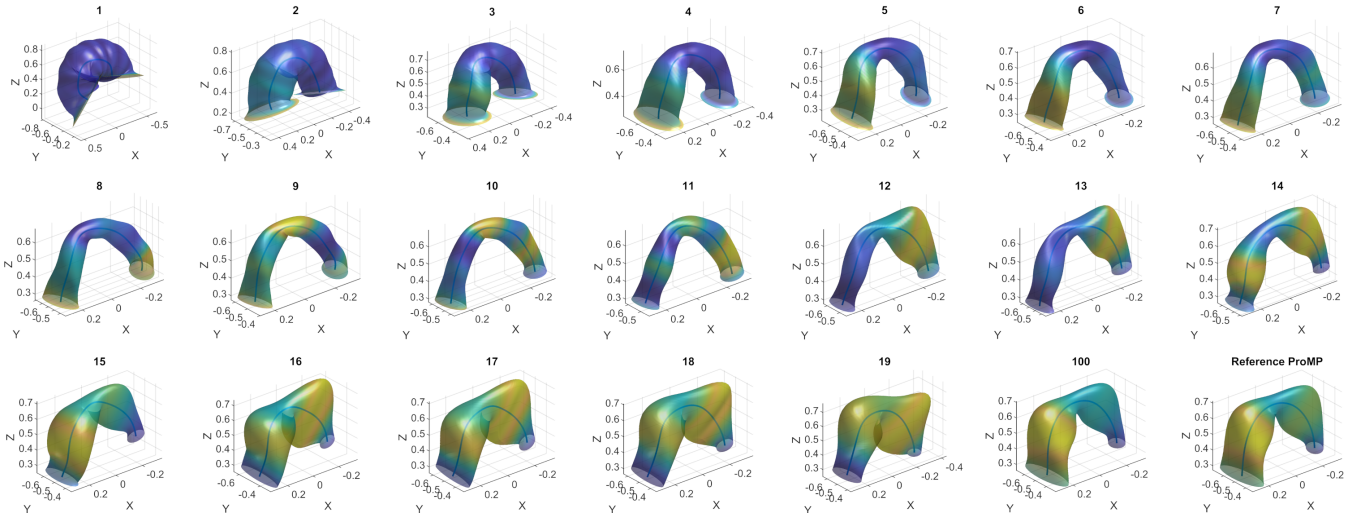


Fig. 5. Evolution of the ProMP during incremental training on the first 19 demonstrations as well as the final ProMP after 100 demonstrations and the reference ProMP for comparison. The blue lines represent the mean $\boldsymbol{\mu}_w$ and the tubes the variance $\boldsymbol{\Sigma}_w$ in terms of two standard deviations. The demonstration number is shown on the top of each plot. During the first few demonstrations, the variance is strongly influenced by the initial value $\boldsymbol{\Sigma}_w = \boldsymbol{I}$. In course of training, the ProMP becomes increasingly similar to the reference.

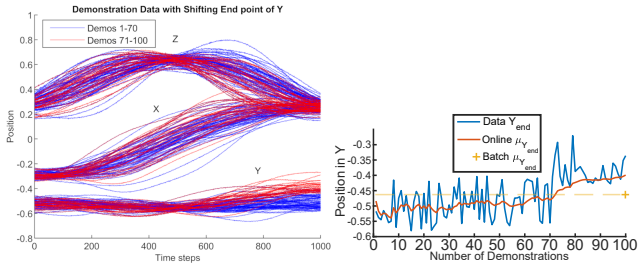


Fig. 6. **Left:** Data set with manipulated order of demonstrations to simulate a change in task constraints. In this case, the endpoint of the trajectory of the Y component is shifted to a new mean. The first 70 demonstrations are shown in **blue**, the last 30 demonstrations, representing the shift in the task constraints, are shown in **red**. **Right:** The endpoint of the trajectory of Y plotted over the number of demonstrations is shown in **Blue**. The shift after demonstration 70 is visible. The **Red** line shows the endpoint of the mean trajectory of the Y component trained with our incremental algorithm. The mean follows the shift in the data. The **Yellow** cross shows the endpoint of the mean trajectory of the Y component trained with the batch EM algorithm. Due to the batch nature of the algorithm, the mean is only computed once after all demonstrations are added. The batch-mean roughly represents the mean of the entire set, or history, of demonstrations and has a rather large distance to the data from demonstration 70 to 100.

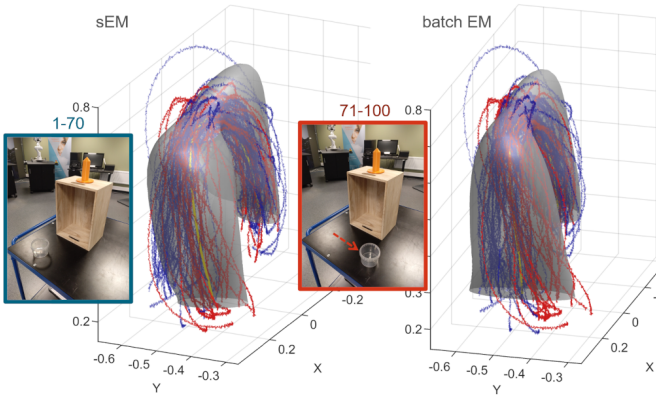


Fig. 7. Comparison of the ProMPs trained in presence of a shift in the task constraints. **Left:** Training result of our online algorithm. **Right:** Training result of the batch EM algorithm. The **blue** lines represent the first 70 demonstrations. The **red** lines represent the last 30 demonstrations with a shifted mean in Y direction at the end of the trajectory. The ProMP mean is shown as a **yellow** line, the variance (two standard deviations) as the **grey** tube. The incrementally trained ProMP **left** represents mostly the recent demonstrations **red**, while the ProMP trained with batch EM represents the entire distribution of demonstrations **blue** and **red**.

$E_F$ of the covariance matrix $\boldsymbol{\Sigma}_w$ shown as a red line in the top right plot of Fig. 4. Apart from the relative error, also the dynamics in the covariance structure indicated by the rotation of the first principal component appear to settle after about 15 demonstrations. The matrix condition number (Fig. 4, bottom left) is actually the lowest, at some points even lower than the reference, during the beginning of the training. This observation can be explained by the initially strong influence of the initial value for $\boldsymbol{\Sigma}_w$ which is chosen to be the identity matrix $\boldsymbol{I}$ with a matrix condition number $\log \kappa(\boldsymbol{I}) = 0$, and the block diagonal prior distribution for $\boldsymbol{\Sigma}_w$. The block diagonal prior suppresses the correlation terms between the the X,Y and Z components of the movement, since it implies the off-diagonal blocks of $\boldsymbol{\Sigma}_w$ to be zero. It follows, that until the influence of the prior is overruled by the data, there is less collinearity in $\boldsymbol{\Sigma}_w$ which could make the matrix near-singular and lead to higher condition numbers.

The variance at the beginning of training is roughly a symmetric tube around the mean which reflects the strong influence of the initial value for the covariance matrices $\boldsymbol{\Sigma}_w = \boldsymbol{\Sigma}_y = \boldsymbol{I}$. This symmetric tube is acceptable at early stages of training since it simply reflects equal uncertainty in all directions. However, the early variance structure can be further improved by use of prior parameter distributions inspired by features of human motion [9].

The estimates of the ProMP mean $\boldsymbol{\mu}_w$ appear to be stable throughout the course of training when we consider its low relative error computed from the Frobenius norm (blue line in Fig. 4 top right). Reaching a near-steady value not before 40 demonstrations, the Bhattacharyya distance (Fig. 4 top

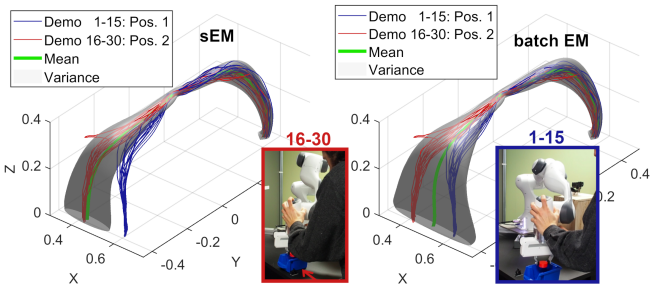


Fig. 8. Comparison of ProMPs trained on unprocessed robot data from a pick-and-place task with a change in the task constraints. **Left:** Training result of our online algorithm. **Right:** Training result of the batch EM algorithm. Axis units are meters. The **blue** lines represent the first 15 demonstrations. The **red** lines represent the last 15 demonstrations with a shifted mean in X direction at the end of the trajectory. The ProMP mean is shown as a **green** line, the variance (two standard deviations) as the **grey** tube. The mean **green** of the incrementally trained **left** is shifted to the most recent demonstrations **red**. The uncertainty about this new mean is still increased, indicated by the relatively large variance **grey**. The ProMP trained with batch EM represents the entire distribution of demonstrations **blue** and **red**, with a mean **green** that ends between the two "place"-positions.

left) indicates slower convergence than the relative errors of the mean and the covariance. However, it is currently unclear how much the Bhattacharyya distance is correlated with the applicability of a ProMP in practice. This relation has to be determined in further experiments involving actual robots.

### C. Discussion on Adaptation to Changes in the Task

The experiment in section III-C had the purpose to demonstrate how our online algorithm can adapt a ProMP to changes in the task constraints or task execution due to the built-in forgetting factor. Looking at the right side of Fig. 6, it is obvious that the incremental learning algorithm adapts the endpoint of the mean trajectory (red line) such as to follow the distribution of the recent demonstrations. This includes the period from demonstration 70 to 100, where the simulated change in the task is happening. Applying the batch EM algorithm, on the other hand, yields a mean endpoint (yellow cross) based on the entire set of demonstrations. This mean endpoint has a rather large distance to the data after the change in the task. The same conclusions can be made by looking at the entire ProMP distributions shown in Fig. 7. On the left, the ProMP trained with our online algorithm has adjusted in terms of mean and variance to the recent data (red) representing the change in the task. On the right, the ProMP trained with the batch algorithm is not particularly adapted to the change in the task, but represents the entire data from before and after the change.

The results from the proof of concept in section IV confirm the results of this experiment on unprocessed robot data. The incrementally trained ProMP is adapted to the shift in the task constraints (Fig. 8, left). The adaptation is especially visible for the ProMP mean. After the change in the task, the variance at the end of the demonstrations is relatively large compared to the actual spread of the data. We would expect the variance to further decrease if more demonstrations after the change in the task were provided and when more of the information of the first position is forgotten. This can be seen in Fig. 7, where 30 demonstrations after the change were provided and the variance represents the spread of the data after the change well. The ProMP trained with the batch algorithm (Fig. 8, right) has a mean that ends between the initial and final "place"-position, which would be a suboptimal reference trajectory to execute this pick-and-place movement on a robot.

We have demonstrated that the forgetting factor in our online algorithm has the desired effect of adapting ProMPs during training. Hence, the basic idea of our learning framework in that the human can shape the robot's motor skill by giving corrective demonstrations can now be realized with our online algorithm. The results imply that our algorithm can incorporate a learning curve of the human as well as changes in the task constraints - both of which have to be expected for pHRC in an unstructured environment. In future work we plan to conduct user studies to evaluate the proposed incremental learning approach with end-users.

## CREDIT AND ACKNOWLEDGEMENT

DS: Conceptualization, Methodology, Investigation, Software, Writing - original draft., MFS, EK: Conceptualization, Writing - review & editing. We thank Johannes Møgster for valuable discussions and help with lab experiments.

## REFERENCES

[1] A. G. Billard, S. Calinon, and R. Dillmann, "Learning from humans," in *Springer Handbook of Robotics*, B. Siciliano and O. Khatib, Eds. Springer International Publishing, 2016, pp. 1995–2014.

[2] B. Nemec, N. Likar, A. Gams, and A. Ude, "Human robot cooperation with compliance adaptation along the motion trajectory," *Autonomous Robots*, vol. 42, no. 5, pp. 1023–1035, 2018.

[3] A. Paraschos, C. Daniel, J. R. Peters, and G. Neumann, "Probabilistic movement primitives," in *Advances in Neural Information Processing Systems 26*. Curran Ass., Inc, 2013, pp. 2616–2624.

[4] M. Ewerton, G. Maeda, J. Peters, and G. Neumann, "Learning motor skills from partially observed movements executed at different speeds," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, W. Burgard, Ed. IEEE, 2015, pp. 456–463.

[5] S. Gomez-Gonzalez, G. Neumann, B. Scholkopf, and J. Peters, "Adaptation and robust learning of probabilistic movement primitives," *IEEE Transactions on Robotics*, vol. 36, no. 2, pp. 366–379, 2020.

[6] A. Paraschos, C. Daniel, J. Peters, and G. Neumann, "Using probabilistic movement primitives in robotics," *Autonomous Robots*, vol. 42, no. 3, pp. 529–551, 2018.

[7] D. Koert, S. Trick, M. Ewerton, M. Lutter, and J. Peters, "Online learning of an open-ended skill library for collaborative tasks," in *2018 IEEE-RAS 18th International Conference on Humanoid Robots (Humanoids)*. IEEE, 2018, pp. 1–9.

[8] A. Conkey and T. Hermans, "Active learning of probabilistic movement primitives," in *2018 IEEE-RAS 19th International Conference on Humanoid Robots (Humanoids)*. IEEE, 2019, pp. 1–8.

[9] D. Schäle, M. F. Stoelen, and E. Kyrkjebø, "Priors inspired by speed-accuracy trade-offs for incremental learning of probabilistic movement primitives," in *Towards Autonomous Robotic Systems*, ser. Lecture Notes in Computer Science. Springer, Cham, 2021.

[10] P. M. Engel and M. R. Heinen, "Incremental learning of multivariate gaussian mixture models," in *Advances in artificial intelligence - SBIA 2010*, ser. LNAI, 0302-9743, A. C. da Rocha Costa, R. M. Vicari, and F. Tonidandel, Eds. Springer, 2010, vol. 6404, pp. 82–91.

[11] D. Lee and C. Ott, "Incremental kinesthetic teaching of motion primitives using the motion refinement tube," *Autonomous Robots*, vol. 31, no. 2-3, pp. 115–131, 2011.

[12] M. Sato and S. Ishii, "On-line em algorithm for the normalized gaussian network," *Neural Comput.*, vol. 12, no. 2, pp. 407–432, 2000.

[13] P. Liang and D. Klein, "Online em for unsupervised models," in *Proceedings of human language technologies: The 2009 annual conf. of the NOAM chapter of the ass. for comp. ling.*, 2009, pp. 611–619.