UNIVERSITY OF LEEDS

Article:
Huang, Y, Rozo, L, Silvério, J et al. (1 more author) (2019) Kernelized movement primitives. The International Journal of Robotics Research, 38 (7). pp. 833-852. ISSN 0278-3649

https://doi.org/10.1177/0278364919846363

Kernelized Movement Primitives

Journal Title
XX(X):1–18

DOI: 10.1177/ToBeAssigned
www.sagepub.com/
SAGE

Yanlong Huang[1], Leonel Rozo[1], João Silvério[1] and Darwin G. Caldwell[1]

Abstract

Imitation learning has been studied widely as a convenient way to transfer human skills to robots. This learning approach is aimed at extracting relevant motion patterns from human demonstrations and subsequently applying these patterns to different situations. Despite many advancements have been achieved, the solutions for coping with unpredicted situations (e.g., obstacles and external perturbations) and high-dimensional inputs are still largely open. In this paper, we propose a novel *kernelized movement primitive* (KMP), which allows the robot to adapt the learned motor skills and fulfill a variety of additional constraints arising over the course of a task. Specifically, KMP is capable of learning trajectories associated with high-dimensional inputs due to the kernel treatment, which in turn renders a model with fewer open parameters in contrast to methods that rely on basis functions. Moreover, we extend our approach by exploiting local trajectory representations in different coordinate systems that describe the task at hand, endowing KMP with reliable extrapolation capabilities in broader domains. We apply KMP to the learning of time-driven trajectories as a special case, where a compact parametric representation describing a trajectory and its first-order derivative is utilized. In order to verify the effectiveness of our method, several examples of trajectory modulations and extrapolations associated with time inputs, as well as trajectory adaptations with high-dimensional inputs are provided.

Keywords

Imitation learning, movement primitives, information theory, kernel-based learning.

1 Introduction

In a myriad of robotic systems, trajectory generation plays a very important role since trajectories govern the robot actions at both joint and task space levels. One popular trajectory generation approach for robots is imitation learning (Ijspeert et al. 2013; Calinon et al. 2007), where the trajectory of interest is learned from human demonstrations. Typically, the learned trajectories can be successfully reproduced or generalized by the robot under conditions that are similar to those in which the demonstrations took place. However, in practice, robots may also encounter unseen situations, such as obstacles or human intervention, which can be considered as new task constraints, requiring the robot to adapt its trajectory in order to perform adequately.

In the context of imitation learning, several algorithms such as dynamic movement primitives (DMP) (Ijspeert et al. 2013) and probabilistic movement primitives (ProMP) (Paraschos et al. 2013) have been developed to generate desired trajectories in various scenarios. However, due to an explicit description of the trajectory dynamics, DMP introduces many open parameters in addition to basis functions and their weighting coefficients. The same problem arises in ProMP, which fits trajectories using basis functions that are manually defined. Moreover, DMP and ProMP were formulated towards the learning of time-driven trajectories (i.e., trajectories explicitly dependent on time), where the learning with high-dimensional inputs was not addressed.

In order to alleviate the modeling of trajectories via specific functions and meanwhile facilitate the learning of trajectories driven by high dimensional inputs, Gaussian mixture model (GMM) (Calinon et al. 2007) has been employed to model the joint distribution of input variables and demonstrated motions. Usually, GMM is complemented with Gaussian mixture regression (GMR) (Cohn et al. 1996) to retrieve a desired trajectory distribution. Despite the improvements with respect to other techniques, adapting learned skills with GMM/GMR is not straightforward. Indeed, it is difficult to re-optimize GMM to fulfill new requirements (e.g., via-points), since this usually requires to re-estimate new model parameters (i.e., mixture coefficients, means and covariance matrices) that actually lie on a high-dimensional space .

An alternative solution to refine trajectories for satisfying new task constraints is reinforcement learning (RL). For instance, a variant of policy improvement with path integrals (Buchli et al. 2011) was employed to optimize the movement pattern of DMP (Stulp and Sigaud 2013). Also, natural actor-critic (Peters et al. 2005) was used to optimize the centers of GMM components (Guenter et al. 2007). However, the time-consuming search of the optimal policy might make the application of RL approaches to on-line refinements (such as those required after perturbations) impractical. In contrast to the RL treatment, ProMP formulates the modulation of trajectories as a Gaussian conditioning problem, and

[1] Department of Advanced Robotics, Istituto Italiano di Tecnologia.

Corresponding author:
Yanlong Huang, Department of Advanced Robotics, Istituto Italiano di Tecnologia, Via Morego 30, 16163 Genova, Italy.
Email: yanlong.huang@iit.it

therefore derives an analytical solution to adapt trajectories towards new via-points or targets. It is worth pointing out that DMP can adapt trajectories towards different goals, however, the via-point constraints are not addressed therein.

Besides the generation of adaptive trajectories, another desired property in imitation learning is extrapolation. Often, human demonstrations are provided for a limited set of task instances, but the robot is expected to apply the learned movement patterns in a wider range of circumstances. In this context, DMP is capable of generating trajectories starting from arbitrary locations and converging to a goal. This is achieved through a formulation based on a spring-damper system whose equilibrium corresponds to the target of the robot motion. In contrast, ProMP and GMM model the distribution of demonstrated trajectories in absolute frames rather than relative frames, which limits their extrapolation capabilities. As an extension of GMM, a task-parameterized formulation is studied in Calinon (2016), which in essence models local (or relative) trajectories and corresponding local patterns, therefore endowing GMM with better extrapolation performance.

While the aforementioned algorithms have achieved reliable performances, we aim for a solution that addresses the most crucial limitations of those approaches. In particular, we propose an algorithm that:

(i) preserves the probabilistic properties exhibited in multiple demonstrations,

(ii) deals with adaptation and superposition of trajectories,

(iii) can be generalized for extrapolations,

(iv) learns human demonstrations associated with high-dimensional inputs while alleviating the need to explicitly define basis functions.

The main contribution of this paper is the development of novel *kernelized movement primitives* (KMPs), which allow us to address the above listed problems using a single framework. Specifically, provided with a distribution of demonstrations, KMP provides a non-parametric solution for imitation learning and hence alleviates the explicit representation of trajectories using basis functions, rendering fewer open parameters and easy implementation. More importantly, in light of the kernel treatment, KMP has the ability to learn demonstrations associated with high-dimensional inputs, which is usually viewed as a non-trivial problem due to the curse of dimensionality.

In addition, this paper extends KMP from a task-parameterized perspective and formulates *local*-KMP, improving the extrapolation capabilities to different task situations described by a set of local coordinate frames. Finally, as a special case, we consider the application of KMP to the learning of time-driven trajectories, which inherits all the advantages of KMP while being suitable for time-scale modulation. For the sake of clear comparison, we list most relevant features of state-of-the-art methods as well as our approach in Table 1. Note that we consider the modulation of robot trajectories to pass through desired via-points and end-points as the adaptation capability.

The structure of this paper is arranged as follows. We formulate imitation learning from an information-theory perspective and propose KMP in Section 2. Subsequently, we extend KMP to deal with trajectory modulation and superposition in Section 3.1 and Section 3.2, respectively. Moreover, we introduce the concept of learning local trajectories into KMP in Section 3.3, augmenting its extrapolation capabilities in task space. In Section 4, we discuss a special application of KMP to time-driven trajectories. We test the performance of KMP on trajectory modulation, superposition and extrapolation in Section 5, where several scenarios are considered, ranging from learning handwritten letters to real robot experiments. After that, we review related work in Section 6. An insightful discussion is provided in Section 7, where we elaborate on the potential of our approach and the similarities between KMP and ProMP, as well as open challenges. Finally, we close with conclusions in Section 8.

Table 1. Comparison Among the State-of-the-Art and KMP

	DMP	ProMP	GMM	Our Approach
Probabilistic	-	✓	✓	✓
Via–point	-	✓	-	✓
End–point	✓	✓	-	✓
Extrapolation	✓	-	-	✓
High-dim Inputs	-	-	✓	✓

2 Kernelized Representation of Movement Trajectories Distribution

Learning from multiple demonstrations allows for encoding trajectory distributions and extracting important or consistent features of the task. In this section, we first illustrate a probabilistic modeling of human demonstrations (Section 2.1), and, subsequently, we exploit the resulting trajectory distribution to derive KMP (Section 2.2).

2.1 Learning from Human Demonstrations

Formally, let us denote the set of demonstrated training data by $\{\{\mathbf{s}_{n,h}, \boldsymbol{\xi}_{n,h}\}_{n=1}^{N}\}_{h=1}^{H}$ where $\mathbf{s}_{n,h} \in \mathbb{R}^{\mathcal{I}}$ is the input and $\boldsymbol{\xi}_{n,h} \in \mathbb{R}^{\mathcal{O}}$ denotes the output. Here, the super-indexes $\mathcal{I}$, $\mathcal{O}$, H and N respectively represent the dimensionality of the input and output space, the number of demonstrations, and the trajectory length. Note that a probabilistic encoding of the demonstrations allows the input $\mathbf{s}$ and output $\boldsymbol{\xi}$ to represent different types of variables. For instance, by considering $\mathbf{s}$ as the position of the robot and $\boldsymbol{\xi}$ as its velocity, the representation becomes an autonomous system. Alternatively, if $\mathbf{s}$ and $\boldsymbol{\xi}$ respectively represent time and position, the resulting encoding corresponds to a time-driven trajectory.

In order to capture the probabilistic distribution of demonstrations, a number of algorithms can be employed, such as GMM (Calinon et al. 2007), hidden Markov models (Rozo et al. 2013), and even a single Gaussian distribution (Englert et al. 2013; Osa et al. 2017), which differ in the type of information that is extracted from the demonstrations. As an example, let us exploit GMM as the model used to encode the training data. More specifically, GMM is employed to estimate the joint probability distribution $\mathcal{P}(\mathbf{s}, \boldsymbol{\xi})$ from

demonstrations, i.e.,

$$\begin{bmatrix}\mathbf{s}\\ \boldsymbol{\xi}\end{bmatrix} \sim \sum_{c=1}^{C} \pi_c \mathcal{N}(\boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c), \tag{1}$$

where π_c, $\boldsymbol{\mu}_c$ and $\boldsymbol{\Sigma}_c$ respectively represent the prior probability, mean and covariance of the c-th Gaussian component, while C denotes the number of Gaussian components.

Furthermore, a probabilistic *reference trajectory* $\{\hat{\boldsymbol{\xi}}_n\}_{n=1}^N$ can be retrieved via GMR (Cohn et al. 1996; Calinon 2016), where each point $\hat{\boldsymbol{\xi}}_n$ associated with $\mathbf{s}_n$ is described by a conditional probability distribution with mean $\hat{\boldsymbol{\mu}}_n$ and covariance $\hat{\boldsymbol{\Sigma}}_n$, i.e, $\hat{\boldsymbol{\xi}}_n|\mathbf{s}_n \sim \mathcal{N}(\hat{\boldsymbol{\mu}}_n, \hat{\boldsymbol{\Sigma}}_n)$ (see Appendix A for details). This probabilistic reference trajectory encapsulates the variability in the demonstrations as well as the correlations among outputs. We take advantage of the probabilistic reference trajectory to derive KMP.

2.2 Kernelized Movement Primitive (KMP)

We start the derivation of KMP by considering a *parametric trajectory*

$$\boldsymbol{\xi}(\mathbf{s}) = \boldsymbol{\Theta}(\mathbf{s})^{\mathsf{T}}\mathbf{w} \tag{2}$$

with the matrix $\boldsymbol{\Theta}(\mathbf{s}) \in \mathbb{R}^{B\mathcal{O}\times\mathcal{O}}$ defined as

$$\boldsymbol{\Theta}(\mathbf{s}) = \begin{bmatrix} \boldsymbol{\varphi}(\mathbf{s}) & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \boldsymbol{\varphi}(\mathbf{s}) & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \boldsymbol{\varphi}(\mathbf{s}) \end{bmatrix}, \tag{3}$$

and the weight vector $\mathbf{w} \in \mathbb{R}^{B\mathcal{O}}$, where $\boldsymbol{\varphi}(\mathbf{s}) \in \mathbb{R}^B$ denotes B-dimensional basis functions*. Furthermore, we assume that $\mathbf{w}$ is normally distributed, i.e., $\mathbf{w} \sim \mathcal{N}(\boldsymbol{\mu}_w, \boldsymbol{\Sigma}_w)$, where the mean $\boldsymbol{\mu}_w$ and the covariance $\boldsymbol{\Sigma}_w$ are *unknown*. Therefore, the parametric trajectory satisfies

$$\boldsymbol{\xi}(\mathbf{s}) \sim \mathcal{N}\left(\boldsymbol{\Theta}(\mathbf{s})^{\mathsf{T}}\boldsymbol{\mu}_w, \boldsymbol{\Theta}(\mathbf{s})^{\mathsf{T}}\boldsymbol{\Sigma}_w\boldsymbol{\Theta}(\mathbf{s})\right). \tag{4}$$

Note that our goal is to imitate the probabilistic reference trajectory $\{\hat{\boldsymbol{\xi}}_n\}_{n=1}^N$, thus we aim to match the parametric trajectory distribution formulated by (4) with the reference trajectory distribution. In order to address this problem, we propose to minimize the *Kullback-Leibler* divergence (KL-divergence) (Kullback and Leibler 1951; Rasmussen and Williams 2006) between both trajectory distributions (Section 2.2.1). Subsequently, we derive optimal solutions for both $\boldsymbol{\mu}_w$ and $\boldsymbol{\Sigma}_w$, and formulate KMP by using the kernel trick in Sections 2.2.2 and 2.2.3, respectively.

2.2.1 Imitation Learning Based on Information Theory:

Since the well-known KL-divergence can be used to measure the distance between two probability distributions, we here exploit it to optimize the parametric trajectory distribution so that it matches the reference trajectory distribution. From the perspective of information transmission, the minimization of KL-divergence guarantees minimal information-loss in the process of imitation learning.

Formally, we consider the minimization of the objective function

$$J_{ini}(\boldsymbol{\mu}_w, \boldsymbol{\Sigma}_w) = \sum_{n=1}^{N} D_{KL}\left(\mathcal{P}_{\mathbf{p}}(\boldsymbol{\xi}|\mathbf{s}_n)||\mathcal{P}_{\mathbf{r}}(\boldsymbol{\xi}|\mathbf{s}_n)\right), \tag{5}$$

where $\mathcal{P}_{\mathbf{p}}(\boldsymbol{\xi}|\mathbf{s}_n)$ represents the probability distribution of the parametric trajectory (4) given the input $\mathbf{s}_n$, i.e.,

$$\mathcal{P}_{\mathbf{p}}(\boldsymbol{\xi}|\mathbf{s}_n) = \mathcal{N}\left(\boldsymbol{\xi}|\boldsymbol{\Theta}(\mathbf{s}_n)^{\mathsf{T}}\boldsymbol{\mu}_w, \boldsymbol{\Theta}(\mathbf{s}_n)^{\mathsf{T}}\boldsymbol{\Sigma}_w\boldsymbol{\Theta}(\mathbf{s}_n)\right), \tag{6}$$

and $\mathcal{P}_{\mathbf{r}}(\boldsymbol{\xi}|\mathbf{s}_n)$ corresponds to the probability distribution of the reference trajectory associated with $\mathbf{s}_n$ (as described in Section 2.1), namely

$$\mathcal{P}_{\mathbf{r}}(\boldsymbol{\xi}|\mathbf{s}_n) = \mathcal{N}(\boldsymbol{\xi}|\hat{\boldsymbol{\mu}}_n, \hat{\boldsymbol{\Sigma}}_n). \tag{7}$$

$D_{KL}(\cdot||\cdot)$ denotes the KL-divergence between the probability distributions $\mathcal{P}_{\mathbf{p}}$ and $\mathcal{P}_{\mathbf{r}}$, which is defined by

$$\begin{aligned} &D_{KL}(\mathcal{P}_{\mathbf{p}}(\boldsymbol{\xi}|\mathbf{s}_n)||\mathcal{P}_{\mathbf{r}}(\boldsymbol{\xi}|\mathbf{s}_n)) \\ &\quad = \int \mathcal{P}_{\mathbf{p}}(\boldsymbol{\xi}|\mathbf{s}_n) \log \frac{\mathcal{P}_{\mathbf{p}}(\boldsymbol{\xi}|\mathbf{s}_n)}{\mathcal{P}_{\mathbf{r}}(\boldsymbol{\xi}|\mathbf{s}_n)} d\boldsymbol{\xi}. \end{aligned} \tag{8}$$

By using the properties of KL-divergence between two Gaussian distributions, we rewrite (5) as

$$\begin{aligned} J_{ini}(\boldsymbol{\mu}_w, \boldsymbol{\Sigma}_w) = \sum_{n=1}^{N} \frac{1}{2} \Big(&\log|\hat{\boldsymbol{\Sigma}}_n| - \log|\boldsymbol{\Theta}(\mathbf{s}_n)^{\mathsf{T}}\boldsymbol{\Sigma}_w\boldsymbol{\Theta}(\mathbf{s}_n)| \\ &- \mathcal{O} + \mathrm{Tr}(\hat{\boldsymbol{\Sigma}}_n^{-1}\boldsymbol{\Theta}(\mathbf{s}_n)^{\mathsf{T}}\boldsymbol{\Sigma}_w\boldsymbol{\Theta}(\mathbf{s}_n)) \\ &+ (\boldsymbol{\Theta}(\mathbf{s}_n)^{\mathsf{T}}\boldsymbol{\mu}_w - \hat{\boldsymbol{\mu}}_n)^{\mathsf{T}}\hat{\boldsymbol{\Sigma}}_n^{-1}(\boldsymbol{\Theta}(\mathbf{s}_n)^{\mathsf{T}}\boldsymbol{\mu}_w - \hat{\boldsymbol{\mu}}_n) \Big), \end{aligned} \tag{9}$$

where $|\cdot|$ and $\mathrm{Tr}(\cdot)$ denote the determinant and trace of a matrix, respectively.

After removing the coefficient '$\frac{1}{2}$', the constant terms $\log|\hat{\boldsymbol{\Sigma}}_n|$ and $\mathcal{O}$, this objective function (9) can be further decomposed into a *mean minimization subproblem* and a *covariance minimization subproblem*. The former is defined by minimizing

$$J_{ini}(\boldsymbol{\mu}_w) = \sum_{n=1}^{N} (\boldsymbol{\Theta}(\mathbf{s}_n)^{\mathsf{T}}\boldsymbol{\mu}_w - \hat{\boldsymbol{\mu}}_n)^{\mathsf{T}}\hat{\boldsymbol{\Sigma}}_n^{-1}(\boldsymbol{\Theta}(\mathbf{s}_n)^{\mathsf{T}}\boldsymbol{\mu}_w - \hat{\boldsymbol{\mu}}_n) \tag{10}$$

and the latter is written as the minimization of

$$\begin{aligned} J_{ini}(\boldsymbol{\Sigma}_w) = \sum_{n=1}^{N} \Big(&- \log|\boldsymbol{\Theta}(\mathbf{s}_n)^{\mathsf{T}}\boldsymbol{\Sigma}_w\boldsymbol{\Theta}(\mathbf{s}_n)| \\ &+ \mathrm{Tr}(\hat{\boldsymbol{\Sigma}}_n^{-1}\boldsymbol{\Theta}(\mathbf{s}_n)^{\mathsf{T}}\boldsymbol{\Sigma}_w\boldsymbol{\Theta}(\mathbf{s}_n)) \Big) \end{aligned}. \tag{11}$$

In the following two sections, we separately solve the mean and covariance subproblems, resulting in the KMP formulation.

2.2.2 Mean Prediction of KMP:

In contrast to kernel ridge regression (KRR) (Saunders et al. 1998; Murphy 2012), we introduce a penalty term $||\boldsymbol{\mu}_w||^2$ into the mean minimization subproblem (10) so as to circumvent the over-fitting problem. Thus, the new mean minimization subproblem can be re-rewritten as

$$\begin{aligned} J(\boldsymbol{\mu}_w) = \sum_{n=1}^{N} &(\boldsymbol{\Theta}(\mathbf{s}_n)^{\mathsf{T}}\boldsymbol{\mu}_w - \hat{\boldsymbol{\mu}}_n)^{\mathsf{T}}\hat{\boldsymbol{\Sigma}}_n^{-1}(\boldsymbol{\Theta}(\mathbf{s}_n)^{\mathsf{T}}\boldsymbol{\mu}_w - \hat{\boldsymbol{\mu}}_n) \\ &+ \lambda\boldsymbol{\mu}_w^{\mathsf{T}}\boldsymbol{\mu}_w, \end{aligned} \tag{12}$$

*The treatment of fitting trajectories by using basis functions has also been studied in DMP (Ijspeert et al. 2013) and ProMP (Paraschos et al. 2013).

where $\lambda > 0$.

The cost function (12) resembles a weighted least squares formulation, except for the penalty term $\lambda \boldsymbol{\mu}_w^{\mathsf{T}} \boldsymbol{\mu}_w$. Also, it is similar to the common quadratic cost function minimized in KRR, where $\hat{\boldsymbol{\Sigma}}_n^{-1} = \mathbf{I}_{\mathcal{O}}$. However, the variability of the demonstrations encapsulated in $\hat{\boldsymbol{\Sigma}}_n$ is introduced in (12) as an importance measure associated to each trajectory datapoint, which can be understood as relaxing or reinforcing the optimization for a particular datapoint. In other words, this covariance-weighted cost function permits large deviations from the reference trajectory points with high covariances, while demanding to be close when the associated covariance is low.

By taking advantage of the dual transformation of KRR, the optimal solution $\boldsymbol{\mu}_w^*$ of (12) can be derived as (see Murphy (2012); Kober et al. (2011) for details)

$$\boldsymbol{\mu}_w^* = \boldsymbol{\Phi}(\boldsymbol{\Phi}^{\mathsf{T}}\boldsymbol{\Phi} + \lambda\boldsymbol{\Sigma})^{-1}\boldsymbol{\mu}, \tag{13}$$

where

$$\begin{aligned} \boldsymbol{\Phi} &= [\boldsymbol{\Theta}(\mathbf{s}_1)\ \boldsymbol{\Theta}(\mathbf{s}_2)\ \cdots\ \boldsymbol{\Theta}(\mathbf{s}_N)], \\ \boldsymbol{\Sigma} &= \text{blockdiag}(\hat{\boldsymbol{\Sigma}}_1,\ \hat{\boldsymbol{\Sigma}}_2,\ \ldots,\ \hat{\boldsymbol{\Sigma}}_N), \\ \boldsymbol{\mu} &= [\hat{\boldsymbol{\mu}}_1^{\mathsf{T}}\ \hat{\boldsymbol{\mu}}_2^{\mathsf{T}}\ \cdots\ \hat{\boldsymbol{\mu}}_N^{\mathsf{T}}]^{\mathsf{T}}. \end{aligned} \tag{14}$$

Subsequently, for a query $\mathbf{s}^*$ (i.e., new input), its corresponding output (expected value) is computed as

$$\mathbb{E}(\boldsymbol{\xi}(\mathbf{s}^*)) = \boldsymbol{\Theta}(\mathbf{s}^*)^{\mathsf{T}}\boldsymbol{\mu}_w^* = \boldsymbol{\Theta}(\mathbf{s}^*)^{\mathsf{T}}\boldsymbol{\Phi}(\boldsymbol{\Phi}^{\mathsf{T}}\boldsymbol{\Phi}+\lambda\boldsymbol{\Sigma})^{-1}\boldsymbol{\mu}. \tag{15}$$

In order to facilitate the application of (15) (particularly for high-dimensional $\mathbf{s}$), we propose to kernelize (15) so as to avoid the explicit definition of basis functions. Let us define the inner product for $\boldsymbol{\varphi}(\mathbf{s}_i)$ and $\boldsymbol{\varphi}(\mathbf{s}_j)$ as

$$\boldsymbol{\varphi}(\mathbf{s}_i)^{\mathsf{T}}\boldsymbol{\varphi}(\mathbf{s}_j) = k(\mathbf{s}_i, \mathbf{s}_j), \tag{16}$$

where $k(\cdot,\cdot)$ is a kernel function. Then, based on (3) and (16), we have

$$\boldsymbol{\Theta}(\mathbf{s}_i)^{\mathsf{T}}\boldsymbol{\Theta}(\mathbf{s}_j) = \begin{bmatrix} k(\mathbf{s}_i,\mathbf{s}_j) & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & k(\mathbf{s}_i,\mathbf{s}_j) & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & k(\mathbf{s}_i,\mathbf{s}_j) \end{bmatrix}, \tag{17}$$

which can be further rewritten as a kernel matrix

$$\mathbf{k}(\mathbf{s}_i,\ \mathbf{s}_j) = \boldsymbol{\Theta}(\mathbf{s}_i)^{\mathsf{T}}\boldsymbol{\Theta}(\mathbf{s}_j) = k(\mathbf{s}_i,\mathbf{s}_j)\mathbf{I}_{\mathcal{O}}, \tag{18}$$

where $\mathbf{I}_{\mathcal{O}}$ is the $\mathcal{O}$-dimensional identity matrix. Also, let us denote the matrix $\mathbf{K}$ as

$$\mathbf{K} = \begin{bmatrix} \mathbf{k}(\mathbf{s}_1,\mathbf{s}_1) & \mathbf{k}(\mathbf{s}_1,\mathbf{s}_2) & \cdots & \mathbf{k}(\mathbf{s}_1,\mathbf{s}_N) \\ \mathbf{k}(\mathbf{s}_2,\mathbf{s}_1) & \mathbf{k}(\mathbf{s}_2,\mathbf{s}_2) & \cdots & \mathbf{k}(\mathbf{s}_2,\mathbf{s}_N) \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{k}(\mathbf{s}_N,\mathbf{s}_1) & \mathbf{k}(\mathbf{s}_N,\mathbf{s}_2) & \cdots & \mathbf{k}(\mathbf{s}_N,\mathbf{s}_N) \end{bmatrix}, \tag{19}$$

and write the matrix $\mathbf{k}^*$ as

$$\mathbf{k}^* = [\mathbf{k}(\mathbf{s}^*,\mathbf{s}_1)\ \mathbf{k}(\mathbf{s}^*,\mathbf{s}_2)\ \cdots\ \mathbf{k}(\mathbf{s}^*,\mathbf{s}_N)], \tag{20}$$

then the prediction in (15) becomes

$$\mathbb{E}(\boldsymbol{\xi}(\mathbf{s}^*)) = \mathbf{k}^*(\mathbf{K} + \lambda\boldsymbol{\Sigma})^{-1}\boldsymbol{\mu}. \tag{21}$$

Note that a similar result was derived in the context of reinforcement learning (Kober et al. 2011) (called cost regularized kernel regression, CrKR). In contrast to the mean prediction of KMP, CrKR models target components separately without considering their correlations, i.e., a diagonal weighted matrix $\mathbf{R}_n = r_n\mathbf{I}_{\mathcal{O}}$ is used instead of the full covariance matrix $\hat{\boldsymbol{\Sigma}}_n^{-1}$ from (12). Furthermore, for the case in which $\hat{\boldsymbol{\Sigma}}_n = \mathbf{I}_{\mathcal{O}}$, the prediction in (21) is identical to the mean of the Gaussian process regression (GPR) (Rasmussen and Williams 2006).

It is worth pointing out that the initial mean minimization subproblem (10) is essentially equivalent to the problem of maximizing the posterior $\prod_{n=1}^{N}\mathcal{P}(\boldsymbol{\Theta}(\mathbf{s}_n)^{\mathsf{T}}\boldsymbol{\mu}_w|\hat{\boldsymbol{\mu}}_n, \hat{\boldsymbol{\Sigma}}_n)$, please refer to Appendix B for the proof. Thus, the optimal solution $\boldsymbol{\mu}_w^*$ can be viewed as the best estimation given the observed reference trajectory distribution.

2.2.3 Covariance Prediction of KMP: Similar to the treatment in (12), we propose to add a penalty term into the covariance minimization subproblem (11) in order to bound the covariance $\boldsymbol{\Theta}(\mathbf{s}_n)^{\mathsf{T}}\boldsymbol{\Sigma}_w\boldsymbol{\Theta}(\mathbf{s}_n)$. On the basis of the properties of the *Rayleigh quotient*, the penalty term could be defined by the largest eigenvalue of $\boldsymbol{\Sigma}_w$. For the sake of easy derivation, we impose a relaxed penalty term $\text{Tr}(\boldsymbol{\Sigma}_w)$ which is larger than the largest eigenvalue of $\boldsymbol{\Sigma}_w$ since $\boldsymbol{\Sigma}_w$ is positive definite. Therefore, the new covariance minimization subproblem becomes

$$\begin{aligned} J(\boldsymbol{\Sigma}_w) = \sum_{n=1}^{N}\Big(&- \log|\boldsymbol{\Theta}(\mathbf{s}_n)^{\mathsf{T}}\boldsymbol{\Sigma}_w\boldsymbol{\Theta}(\mathbf{s}_n)| \\ &+ \text{Tr}(\hat{\boldsymbol{\Sigma}}_n^{-1}\boldsymbol{\Theta}(\mathbf{s}_n)^{\mathsf{T}}\boldsymbol{\Sigma}_w\boldsymbol{\Theta}(\mathbf{s}_n))\Big) + \lambda_c\text{Tr}(\boldsymbol{\Sigma}_w) \end{aligned} \tag{22}$$

with $\lambda_c > 0$. By computing the derivative of (22) with respect to $\boldsymbol{\Sigma}_w$ and setting it to 0, we have[†]

$$\sum_{n=1}^{N}\Big(-\boldsymbol{\Sigma}_w^{-1} + \boldsymbol{\Theta}(\mathbf{s}_n)\hat{\boldsymbol{\Sigma}}_n^{-1}\boldsymbol{\Theta}(\mathbf{s}_n)^{\mathsf{T}}\Big) + \lambda_c\mathbf{I} = 0. \tag{23}$$

Furthermore, we can rewrite (23) in a compact form by using $\boldsymbol{\Phi}$ and $\boldsymbol{\Sigma}$ from (14) and derive the optimal solution $\boldsymbol{\Sigma}_w^*$ as

$$\boldsymbol{\Sigma}_w^* = N(\boldsymbol{\Phi}\boldsymbol{\Sigma}^{-1}\boldsymbol{\Phi}^{\mathsf{T}} + \lambda_c\mathbf{I})^{-1}. \tag{24}$$

This solution resembles the covariance of weighted least square estimation, except for the factor 'N' and the regularized term $\lambda_c\mathbf{I}$.

According to the *Woodbury identity*[‡], we can determine the covariance of $\boldsymbol{\xi}(\mathbf{s}^*)$ for a query $\mathbf{s}^*$ as

$$\begin{aligned} \mathbb{D}(\boldsymbol{\xi}(\mathbf{s}^*)) &= \boldsymbol{\Theta}(\mathbf{s}^*)^{\mathsf{T}}\boldsymbol{\Sigma}_w^*\boldsymbol{\Theta}(\mathbf{s}^*) \\ &= N\boldsymbol{\Theta}(\mathbf{s}^*)^{\mathsf{T}}(\boldsymbol{\Phi}\boldsymbol{\Sigma}^{-1}\boldsymbol{\Phi}^{\mathsf{T}} + \lambda_c\mathbf{I})^{-1}\boldsymbol{\Theta}(\mathbf{s}^*) \\ &= \frac{N}{\lambda_c}\boldsymbol{\Theta}(\mathbf{s}^*)^{\mathsf{T}}\left(\mathbf{I} - \boldsymbol{\Phi}(\boldsymbol{\Phi}^{\mathsf{T}}\boldsymbol{\Phi}+\lambda_c\boldsymbol{\Sigma})^{-1}\boldsymbol{\Phi}^{\mathsf{T}}\right)\boldsymbol{\Theta}(\mathbf{s}^*). \end{aligned} \tag{25}$$

Recall that we defined the kernel matrix in (18)-(19), and hence the covariance of $\boldsymbol{\xi}(\mathbf{s}^*)$ becomes

$$\mathbb{D}(\boldsymbol{\xi}(\mathbf{s}^*)) = \frac{N}{\lambda_c}\left(\mathbf{k}(\mathbf{s}^*,\mathbf{s}^*) - \mathbf{k}^*(\mathbf{K} + \lambda_c\boldsymbol{\Sigma})^{-1}\mathbf{k}^{*\mathsf{T}}\right). \tag{26}$$

[†]The following results on matrix derivatives (Petersen and Pedersen 2008) are used: $\frac{\partial|\mathbf{AXB}|}{\partial\mathbf{X}} = |\mathbf{AXB}|(\mathbf{X}^{\mathsf{T}})^{-1}$ and $\frac{\partial}{\partial\mathbf{X}}\text{Tr}(\mathbf{AXB}) = \mathbf{A}^{\mathsf{T}}\mathbf{B}^{\mathsf{T}}$.

[‡]$(\mathbf{A}+\mathbf{CBC}^{\mathsf{T}})^{-1} = \mathbf{A}^{-1} - \mathbf{A}^{-1}\mathbf{C}(\mathbf{B}^{-1} + \mathbf{C}^{\mathsf{T}}\mathbf{A}^{-1}\mathbf{C})^{-1}\mathbf{C}^{\mathsf{T}}\mathbf{A}^{-1}$.

Algorithm 1 *Kernelized Movement Primitive*

1: **Initialization**
- Define the kernel $k(\cdot,\cdot)$ and set the factors λ and λ_c.

2: **Learning from demonstrations** (see Section 2.1)
- Collect demonstrations $\{\{\mathbf{s}_{n,h}, \boldsymbol{\xi}_{n,h}\}_{n=1}^{N}\}_{h=1}^{H}$.
- Extract the reference database $\{\mathbf{s}_n, \hat{\boldsymbol{\mu}}_n, \hat{\boldsymbol{\Sigma}}_n\}_{n=1}^{N}$.

3: **Prediction using KMP** (see Section 2.2)
- *Input*: query $\mathbf{s}^*$.
- Calculate $\boldsymbol{\Sigma}$, $\boldsymbol{\mu}$, $\mathbf{K}$ and $\mathbf{k}^*$ using (14), (19) and (20).
- *Output*: $\mathbb{E}(\boldsymbol{\xi}(\mathbf{s}^*)) = \mathbf{k}^*(\mathbf{K}+\lambda\boldsymbol{\Sigma})^{-1}\boldsymbol{\mu}$ and $\mathbb{D}(\boldsymbol{\xi}(\mathbf{s}^*)) = \frac{N}{\lambda_c}\left(\mathbf{k}(\mathbf{s}^*,\mathbf{s}^*) - \mathbf{k}^*(\mathbf{K}+\lambda_c\boldsymbol{\Sigma})^{-1}\mathbf{k}^{*\mathsf{T}}\right)$.

In addition to the factor '$\frac{N}{\lambda_c}$', the covariance formula in (26) differs from the covariances defined in GPR and CrKR in two essential aspects. First, the variability $\boldsymbol{\Sigma}$ extracted from demonstrations (as defined in (14)) is used in the term $(\mathbf{K}+\lambda_c\boldsymbol{\Sigma})^{-1}$, while the identity matrix and the diagonal weighted matrix are used in GPR and CrKR, respectively. Second, in contrast to the diagonal covariances predicted by GPR and CrKR, KMP predicts a full matrix covariance which allows for predicting the correlations between output components. For the purpose of convenient descriptions in the following discussion, we refer to $\mathbf{D} = \{\mathbf{s}_n, \hat{\boldsymbol{\mu}}_n, \hat{\boldsymbol{\Sigma}}_n\}_{n=1}^{N}$ as the *reference database*. The prediction of both the mean and covariance using KMP is summarized in Algorithm 1.

3 Extensions of Kernelized Movement Primitive

As previously explained, human demonstrations can be used to retrieve a distribution of trajectories that the robot exploits to carry out a specific task. However, in dynamic and unstructured environments the robot also needs to adapt its motions when required. For example, if an obstacle suddenly occupies an area that intersects the robot motion path, the robot needs to modulate its movement trajectory so that collisions are avoided. A similar modulation is necessary (e.g., in pick-and-place and reaching tasks) when the target varies its location during the task execution. The trajectory modulation problem will be addressed in Section 3.1 by exploiting the proposed KMP formulation.

Besides the modulation of a single trajectory, another challenging problem arises when the robot is given a set of candidate trajectories to follow, which represent feasible solutions for the task. Each of them may be assigned with a different priority (extracted, for example, from task constraints). These candidate trajectories can be exploited to compute a mixed trajectory so as to balance all the feasible solutions according to their priorities. We cope with the superposition problem in Section 3.2 by using KMP.

Finally, human demonstrations are often provided in a relatively convenient task space. However, the robot might be expected to apply the learned skill to a broader domain. In order to address this problem, we extend KMP by using local coordinate systems and affine transformations as in Calinon (2016); Huang et al. (2018), which allows KMP to exhibit better extrapolation capabilities (Section 3.3).

3.1 Trajectory Modulation Using KMP

We here consider trajectory modulation in terms of adapting trajectories to pass through new via-points/end-points. Formally, let us define M new desired points as $\{\bar{\mathbf{s}}_m, \bar{\boldsymbol{\xi}}_m\}_{m=1}^{M}$ associated with conditional probability distributions $\bar{\boldsymbol{\xi}}_m|\bar{\mathbf{s}}_m \sim \mathcal{N}(\bar{\boldsymbol{\mu}}_m, \bar{\boldsymbol{\Sigma}}_m)$. These conditional distributions can be designed based on new task requirements. For instance, if there are new via-points that the robot needs to pass through with high precision, small covariances $\bar{\boldsymbol{\Sigma}}_m$ are assigned. On the contrary, for via-points that allow for large tracking errors, high covariances can be set.

In order to consider both new desired points and the reference trajectory distribution simultaneously, we reformulate the original objective function defined in (5) as

$$J_{ini}^{U}(\boldsymbol{\mu}_w,\boldsymbol{\Sigma}_w)=\sum_{n=1}^{N} D_{KL}\Big(\mathcal{P}_{\mathbf{p}}(\boldsymbol{\xi}|\mathbf{s}_n)||\mathcal{P}_{\mathbf{r}}(\boldsymbol{\xi}|\mathbf{s}_n)\Big) + \sum_{m=1}^{M} D_{KL}\Big(\mathcal{P}_{\mathbf{p}}(\boldsymbol{\xi}|\bar{\mathbf{s}}_m)||\mathcal{P}_{\mathbf{d}}(\boldsymbol{\xi}|\bar{\mathbf{s}}_m)\Big) \tag{27}$$

with

$$\mathcal{P}_{\mathbf{p}}(\boldsymbol{\xi}|\bar{\mathbf{s}}_m)=\mathcal{N}\big(\boldsymbol{\xi}|\boldsymbol{\Theta}(\bar{\mathbf{s}}_m)^{\mathsf{T}}\boldsymbol{\mu}_w,\boldsymbol{\Theta}(\bar{\mathbf{s}}_m)^{\mathsf{T}}\boldsymbol{\Sigma}_w\boldsymbol{\Theta}(\bar{\mathbf{s}}_m)\big) \tag{28}$$

and

$$\mathcal{P}_{\mathbf{d}}(\boldsymbol{\xi}|\bar{\mathbf{s}}_m) = \mathcal{N}(\boldsymbol{\xi}|\bar{\boldsymbol{\mu}}_m, \bar{\boldsymbol{\Sigma}}_m). \tag{29}$$

Let $\bar{\mathbf{D}} = \{\bar{\mathbf{s}}_m, \bar{\boldsymbol{\mu}}_m, \bar{\boldsymbol{\Sigma}}_m\}_{m=1}^{M}$ denote the *desired database*. We can concatenate the reference database $\mathbf{D}$ with the desired database $\bar{\mathbf{D}}$ and generate an *extended reference database* $\{\mathbf{s}_i^U, \boldsymbol{\mu}_i^U, \boldsymbol{\Sigma}_i^U\}_{i=1}^{N+M}$, which is defined as follows

$$\begin{cases} \mathbf{s}_i^U=\mathbf{s}_i, \quad \boldsymbol{\mu}_i^U=\hat{\boldsymbol{\mu}}_i, \quad \boldsymbol{\Sigma}_i^U=\hat{\boldsymbol{\Sigma}}_i, & \text{if} \quad 1\le i\le N \\ \mathbf{s}_i^U=\bar{\mathbf{s}}_{i-N}, \boldsymbol{\mu}_i^U=\bar{\boldsymbol{\mu}}_{i-N}, \boldsymbol{\Sigma}_i^U=\bar{\boldsymbol{\Sigma}}_{i-N}, & \text{if} \quad N<i\le N+M \end{cases}. \tag{30}$$

Then, the objective function (27) can be written as follows

$$J_{ini}^{U}(\boldsymbol{\mu}_w,\boldsymbol{\Sigma}_w)=\sum_{i=1}^{M+N} D_{KL}\Big(\mathcal{P}_{\mathbf{p}}(\boldsymbol{\xi}|\mathbf{s}_i^U)||\mathcal{P}_{\mathbf{u}}(\boldsymbol{\xi}|\mathbf{s}_i^U)\Big), \tag{31}$$

with

$$\mathcal{P}_{\mathbf{p}}(\boldsymbol{\xi}|\mathbf{s}_i^U)=\mathcal{N}\big(\boldsymbol{\xi}|\boldsymbol{\Theta}(\mathbf{s}_i^U)^{\mathsf{T}}\boldsymbol{\mu}_w,\boldsymbol{\Theta}(\mathbf{s}_i^U)^{\mathsf{T}}\boldsymbol{\Sigma}_w\boldsymbol{\Theta}(\mathbf{s}_i^U)\big) \tag{32}$$

and

$$\mathcal{P}_{\mathbf{u}}(\boldsymbol{\xi}|\mathbf{s}_i^U) = \mathcal{N}(\boldsymbol{\xi}|\boldsymbol{\mu}_i^U, \boldsymbol{\Sigma}_i^U). \tag{33}$$

Note that (31) has the same form as (5). Hence, for the problem of enforcing trajectories to pass through desired via-points/end-points, we can first concatenate the original reference database with the desired database through (30) and, subsequently, with the extended reference database, we follow Algorithm 1 to predict the mean and covariance for new queries $\mathbf{s}^*$.

It is worth pointing out that there might exist conflicts between the desired database and the original reference database. In order to illustrate this issue clearly, let us consider an extreme case: if there exist a new input $\bar{\mathbf{s}}_m = \mathbf{s}_n$, but $\bar{\boldsymbol{\mu}}_m$ is distant from $\hat{\boldsymbol{\mu}}_n$ while $\bar{\boldsymbol{\Sigma}}_m$ and $\hat{\boldsymbol{\Sigma}}_n$ are nearly the same, then the optimal solution of (31) corresponding to the query $\mathbf{s}_n$ can only be a trade-off

between $\bar{\boldsymbol{\mu}}_m$ and $\hat{\boldsymbol{\mu}}_n$. In the context of trajectory modulation using via-points/end-points, it is natural to consider new desired points with the highest preference. Thus, we propose to update the reference database from the perspective of reducing the above mentioned conflicts while maintaining most of datapoints in the reference database. The update procedure is carried out as follows. For each datapoint $\{\bar{\mathbf{s}}_m, \bar{\boldsymbol{\mu}}_m, \bar{\boldsymbol{\Sigma}}_m\}$ in the desired database, we first compare its input $\bar{\mathbf{s}}_m$ with the inputs $\{\mathbf{s}_n\}_{n=1}^N$ of the reference database so as to find the nearest datapoint $\{\mathbf{s}_r, \hat{\boldsymbol{\mu}}_r, \hat{\boldsymbol{\Sigma}}_r\}$ that satisfies $d(\bar{\mathbf{s}}_m, \mathbf{s}_r) \leq d(\bar{\mathbf{s}}_m, \mathbf{s}_n), \forall n \in \{1, 2, \ldots, N\}$, where $d(\cdot)$ could be an arbitrary distance measure such as 2-norm. If the nearest distance $d(\bar{\mathbf{s}}_m, \mathbf{s}_r)$ is smaller than a predefined threshold $\zeta > 0$, we replace $\{\mathbf{s}_r, \hat{\boldsymbol{\mu}}_r, \hat{\boldsymbol{\Sigma}}_r\}$ with $\{\bar{\mathbf{s}}_m, \bar{\boldsymbol{\mu}}_m, \bar{\boldsymbol{\Sigma}}_m\}$; Otherwise, we insert $\{\bar{\mathbf{s}}_m, \bar{\boldsymbol{\mu}}_m, \bar{\boldsymbol{\Sigma}}_m\}$ into the reference database. More specifically, given a new desired point $\{\bar{\mathbf{s}}_m, \bar{\boldsymbol{\xi}}_m\}$ described by $\bar{\boldsymbol{\xi}}_m | \bar{\mathbf{s}}_m \sim \mathcal{N}(\bar{\boldsymbol{\mu}}_m, \bar{\boldsymbol{\Sigma}}_m)$, we update the reference database according to

$$\begin{cases} \mathbf{D} \leftarrow \{\mathbf{D} / \{\mathbf{s}_r, \hat{\boldsymbol{\mu}}_r, \hat{\boldsymbol{\Sigma}}_r\}\} \cup \{\bar{\mathbf{s}}_m, \bar{\boldsymbol{\mu}}_m, \bar{\boldsymbol{\Sigma}}_m\}, & \text{if } d(\bar{\mathbf{s}}_m, \mathbf{s}_r) < \zeta, \\ \mathbf{D} \leftarrow \mathbf{D} \cup \{\bar{\mathbf{s}}_m, \bar{\boldsymbol{\mu}}_m, \bar{\boldsymbol{\Sigma}}_m\}, & \text{otherwise}, \end{cases} \tag{34}$$

where $r = \arg\min_n d(\bar{\mathbf{s}}_m, \mathbf{s}_n), n \in \{1, 2, \ldots, N\}$ and the symbols '/' and '$\cup$' represent exclusion and union operations, respectively.

3.2 Trajectory Superposition Using KMP

In addition to the modulation operations on a single trajectory, we extend KMP to mix multiple trajectories that represent different feasible solutions for a task, with different priorities. Formally, given a set of L reference trajectory distributions, associated with inputs and corresponding priorities $\gamma_{n,l}$, denoted as $\{\{\mathbf{s}_n, \hat{\boldsymbol{\xi}}_{n,l}, \gamma_{n,l}\}_{n=1}^N\}_{l=1}^L$, where $\hat{\boldsymbol{\xi}}_{n,l} | \mathbf{s}_n \sim \mathcal{N}(\hat{\boldsymbol{\mu}}_{n,l}, \hat{\boldsymbol{\Sigma}}_{n,l})$, and $\gamma_{n,l} \in (0, 1)$ is a priority assigned to the point $\{\mathbf{s}_n, \hat{\boldsymbol{\xi}}_{n,l}\}$ satisfying $\sum_{l=1}^L \gamma_{n,l} = 1$.

Since each priority indicates the importance of one datapoint in a reference trajectory, we use them to weigh the information-loss as follows

$$J_{ini}^S(\boldsymbol{\mu}_w, \boldsymbol{\Sigma}_w) = \sum_{n=1}^N \sum_{l=1}^L \gamma_{n,l} D_{KL}\left(\mathcal{P}_{\mathbf{p}}(\boldsymbol{\xi}|\mathbf{s}_n) || \mathcal{P}_{\mathbf{s}}^l(\boldsymbol{\xi}|\mathbf{s}_n)\right), \tag{35}$$

where $\mathcal{P}_{\mathbf{s}}^l$ is defined as

$$\mathcal{P}_{\mathbf{s}}^l(\boldsymbol{\xi}|\mathbf{s}_n) = \mathcal{N}(\boldsymbol{\xi}|\hat{\boldsymbol{\mu}}_{n,l}, \hat{\boldsymbol{\Sigma}}_{n,l}), \tag{36}$$

representing the distribution of the l-th reference trajectory given the input $\mathbf{s}_n$.

Similar to the decomposition in (9)–(11), the objective function (35) can be decomposed into a *weighted mean minimization subproblem* and a *weighted covariance minimization subproblem*. The former is written as

$$J_{ini}^S(\boldsymbol{\mu}_w) = \sum_{n=1}^N \sum_{l=1}^L \gamma_{n,l} (\boldsymbol{\Theta}(\mathbf{s}_n)^\mathsf{T} \boldsymbol{\mu}_w - \hat{\boldsymbol{\mu}}_{n,l})^\mathsf{T} \hat{\boldsymbol{\Sigma}}_{n,l}^{-1} (\boldsymbol{\Theta}(\mathbf{s}_n)^\mathsf{T} \boldsymbol{\mu}_w - \hat{\boldsymbol{\mu}}_{n,l}), \tag{37}$$

and the latter is

$$J_{ini}^S(\boldsymbol{\Sigma}_w) = \sum_{n=1}^N \sum_{l=1}^L \gamma_{n,l} \Big(-\log |\boldsymbol{\Theta}(\mathbf{s}_n)^\mathsf{T} \boldsymbol{\Sigma}_w \boldsymbol{\Theta}(\mathbf{s}_n)| + \mathrm{Tr}(\hat{\boldsymbol{\Sigma}}_{n,l}^{-1} \boldsymbol{\Theta}(\mathbf{s}_n)^\mathsf{T} \boldsymbol{\Sigma}_w \boldsymbol{\Theta}(\mathbf{s}_n)) \Big). \tag{38}$$

It can be proved that the weighted mean subproblem can be solved by minimizing (see Appendix C)

$$\tilde{J}_{ini}^S(\boldsymbol{\mu}_w) = \sum_{n=1}^N (\boldsymbol{\Theta}(\mathbf{s}_n)^\mathsf{T} \boldsymbol{\mu}_w - \boldsymbol{\mu}_n^S)^\mathsf{T} {\boldsymbol{\Sigma}_n^S}^{-1} (\boldsymbol{\Theta}(\mathbf{s}_n)^\mathsf{T} \boldsymbol{\mu}_w - \boldsymbol{\mu}_n^S), \tag{39}$$

and the weighted covariance subproblem is equivalent to the problem of minimizing (see Appendix D)

$$\tilde{J}_{ini}^S(\boldsymbol{\Sigma}_w) = \sum_{n=1}^N \Big(-\log |\boldsymbol{\Theta}(\mathbf{s}_n)^\mathsf{T} \boldsymbol{\Sigma}_w \boldsymbol{\Theta}(\mathbf{s}_n)| + \mathrm{Tr}({\boldsymbol{\Sigma}_n^S}^{-1} \boldsymbol{\Theta}(\mathbf{s}_n)^\mathsf{T} \boldsymbol{\Sigma}_w \boldsymbol{\Theta}(\mathbf{s}_n)) \Big), \tag{40}$$

where

$${\boldsymbol{\Sigma}_n^S}^{-1} = \sum_{l=1}^L \left(\hat{\boldsymbol{\Sigma}}_{n,l} / \gamma_{n,l}\right)^{-1} \quad \text{and} \tag{41}$$

$$\boldsymbol{\mu}_n^S = \boldsymbol{\Sigma}_n^S \sum_{l=1}^L \left(\hat{\boldsymbol{\Sigma}}_{n,l} / \gamma_{n,l}\right)^{-1} \hat{\boldsymbol{\mu}}_{n,l}. \tag{42}$$

Observe that (39) and (40) have the same form as the subproblems defined in (10) and (11), respectively. Note that the definitions in (41) and (42) essentially correspond to the product of L Gaussian distributions $\mathcal{N}(\hat{\boldsymbol{\mu}}_{n,l}, \hat{\boldsymbol{\Sigma}}_{n,l}/\gamma_{n,l})$ with $l = 1, 2, \ldots, L$, given by

$$\mathcal{N}(\boldsymbol{\mu}_n^S, \boldsymbol{\Sigma}_n^S) \propto \prod_{l=1}^L \mathcal{N}(\hat{\boldsymbol{\mu}}_{n,l}, \hat{\boldsymbol{\Sigma}}_{n,l}/\gamma_{n,l}). \tag{43}$$

Thus, for the problem of trajectory superposition, we first determine a *mixed reference database* $\{\mathbf{s}_n, \boldsymbol{\mu}_n^S, \boldsymbol{\Sigma}_n^S\}_{n=1}^N$ through (43), then we employ Algorithm 1 to predict the corresponding mixed trajectory points for arbitrary queries. Note that the weighted mean minimization subproblem (37) can be interpreted as the maximization of the weighted posterior $\prod_{n=1}^N \prod_{l=1}^L \mathcal{P}\left(\boldsymbol{\Theta}(\mathbf{s}_n)^\mathsf{T} \boldsymbol{\mu}_w | \hat{\boldsymbol{\mu}}_{n,l}, \hat{\boldsymbol{\Sigma}}_{n,l}\right)^{\gamma_{n,l}}$. In comparison with the trajectory mixture in ProMP (Paraschos et al. 2013), we here consider an optimization problem with an unknown $\boldsymbol{\mu}_w$ rather than the direct product of a set of known probabilities.

3.3 Local Movement Learning Using KMP

So far we have considered trajectories that are represented with respect to the same global frame (coordinate system). In order to enhance the extrapolation capability of KMP in task space, human demonstrations can be encoded in local frames[§] so as to extract local movement patterns, which can then be applied to a wider range of task instances. Usually, the definition of local frames depends on the task at hand. For example, in a transportation task where the robot moves

[§] Also referred to as task parameters in Calinon (2016).

Algorithm 2 *Local Kernelized Movement Primitives with Via-points/End-points*

1: **Initialization**
- Define $k(\cdot,\cdot)$ and set λ and λ_c.
- Determine P local frames $\{\mathbf{A}^{(p)}, \mathbf{b}^{(p)}\}_{p=1}^{P}$.

2: **Learning from local demonstrations**
- Collect demonstrations $\{\{\mathbf{s}_{n,h}, \boldsymbol{\xi}_{n,h}\}_{n=1}^{N}\}_{h=1}^{H}$ in $\{O\}$.
- Project demonstrations into local frames via (44).
- Extract local reference databases $\{\mathbf{s}_n^{(p)}, \hat{\boldsymbol{\mu}}_n^{(p)}, \hat{\boldsymbol{\Sigma}}_n^{(p)}\}_{n=1}^{N}$.

3: **Update local reference databases**
- Project via-points/end-points into local frames via (44).
- Update local reference databases via (34).
- Update $\mathbf{K}^{(p)}$, $\boldsymbol{\mu}^{(p)}$, $\boldsymbol{\Sigma}^{(p)}$ in each frame $\{p\}$.

4: **Prediction using local-KMPs**
- *Input*: query $\mathbf{s}^*$.
- Update P local frames based on new task requirements.
- Project $\mathbf{s}^*$ into local frames via (44), yielding $\{\mathbf{s}^{*(p)}\}_{p=1}^{P}$.
- Predict the local trajectory point associated with $\mathbf{s}^{*(p)}$ in each frame $\{p\}$ using KMP.
- *Output*: Compute $\widetilde{\boldsymbol{\xi}}(\mathbf{s}^*)$ in the frame $\{O\}$ using (46).

an object from a starting position (that may vary) to different target locations, two local frames can be defined respectively at the starting and ending positions.

Formally, let us define P local frames as $\{\mathbf{A}^{(p)}, \mathbf{b}^{(p)}\}_{p=1}^{P}$, where $\mathbf{A}^{(p)}$ and $\mathbf{b}^{(p)}$ respectively represent the rotation matrix and the translation vector of frame $\{p\}$ with respect to the base frame $\{O\}$. Demonstrations are projected into each frame $\{p\}$, resulting in new trajectory points $\{\{\mathbf{s}_{n,h}^{(p)}, \boldsymbol{\xi}_{n,h}^{(p)}\}_{n=1}^{N}\}_{h=1}^{H}$ for each local frame, where

$$\begin{bmatrix}\mathbf{s}_{n,h}^{(p)}\\ \boldsymbol{\xi}_{n,h}^{(p)}\end{bmatrix} = \begin{bmatrix}\mathbf{A}_s^{(p)} & \mathbf{0}\\ \mathbf{0} & \mathbf{A}_\xi^{(p)}\end{bmatrix}^{-1}\left(\begin{bmatrix}\mathbf{s}_{n,h}\\ \boldsymbol{\xi}_{n,h}\end{bmatrix} - \begin{bmatrix}\mathbf{b}_s^{(p)}\\ \mathbf{b}_\xi^{(p)}\end{bmatrix}\right), \quad (44)$$

with $\mathbf{A}_s^{(p)} = \mathbf{A}_\xi^{(p)} = \mathbf{A}^{(p)}$ and $\mathbf{b}_s^{(p)} = \mathbf{b}_\xi^{(p)} = \mathbf{b}^{(p)}$¶. Subsequently, by following the procedure in Section 2.1, for each local frame $\{p\}$ we can generate a *local reference database* $\mathbf{D}^{(p)} = \{\mathbf{s}_n^{(p)}, \hat{\boldsymbol{\mu}}_n^{(p)}, \hat{\boldsymbol{\Sigma}}_n^{(p)}\}_{n=1}^{N}$.

We refer to the learning of KMPs in local frames as *local*-KMPs. For sake of simplicity, we only discuss the trajectory modulations with via-points/end-points. The operation of trajectory superposition can be treated as explained in Section 3.2. Given a set of desired points in the robot base frame $\{O\}$ described by the desired database $\{\bar{\mathbf{s}}_m, \bar{\boldsymbol{\mu}}_m, \bar{\boldsymbol{\Sigma}}_m\}_{m=1}^{M}$, we project the desired database into local frames using (44), leading to the set of transformed *local desired databases* $\bar{\mathbf{D}}^{(p)} = \{\bar{\mathbf{s}}_m^{(p)}, \bar{\boldsymbol{\mu}}_m^{(p)}, \bar{\boldsymbol{\Sigma}}_m^{(p)}\}_{m=1}^{M}$ with $p = \{1, 2, \ldots, P\}$. Then, we carry out the update procedure described by (34) in each frame $\{p\}$ and obtain a new local reference database $\mathbf{D}^{(p)}$.

For a new input $\mathbf{s}^*$ in the base frame $\{O\}$, we first project it into local frames using the input transformation in (44), yielding local inputs $\{\mathbf{s}^{*(p)}\}_{p=1}^{P}$. Note that, during the prediction phase, local frames might be updated depending on new task requirements and the corresponding task parameters $\mathbf{A}^{(p)}$ and $\mathbf{b}^{(p)}$ might vary accordingly. Later, in each frame $\{p\}$ we can predict a local trajectory point $\widetilde{\boldsymbol{\xi}}^{(p)}(\mathbf{s}^{*(p)}) \sim \mathcal{N}(\boldsymbol{\mu}^{*(p)}, \boldsymbol{\Sigma}^{*(p)})$ with updated mean $\boldsymbol{\mu}^{*(p)}$ and covariance $\boldsymbol{\Sigma}^{*(p)}$ by using (21) and (26). Furthermore, new local trajectory points from all local frames can be simultaneously transformed into the robot base frame using an inverse formulation of (44). Thus, for the query $\mathbf{s}^*$ in the base frame $\{O\}$, its corresponding trajectory point $\widetilde{\boldsymbol{\xi}}(\mathbf{s}^*)$ in $\{O\}$ can be determined by maximizing the product of linearly transformed Gaussian distributions

$$\widetilde{\boldsymbol{\xi}}(\mathbf{s}^*) = \arg\max_{\boldsymbol{\xi}} \prod_{p=1}^{P} \mathcal{N}\Big(\boldsymbol{\xi} \,|\, \underbrace{\mathbf{A}_\xi^{(p)}\boldsymbol{\mu}^{*(p)} + \mathbf{b}_\xi^{(p)}}_{\widetilde{\boldsymbol{\mu}}_p}, \underbrace{\mathbf{A}_\xi^{(p)}\boldsymbol{\Sigma}^{*(p)}\mathbf{A}_\xi^{(p)\mathsf{T}}}_{\widetilde{\boldsymbol{\Sigma}}_p}\Big), \quad (45)$$

whose optimal solution is

$$\widetilde{\boldsymbol{\xi}}(\mathbf{s}^*) = \Big(\sum_{p=1}^{P}\widetilde{\boldsymbol{\Sigma}}_p^{-1}\Big)^{-1}\sum_{p=1}^{P}\widetilde{\boldsymbol{\Sigma}}_p^{-1}\widetilde{\boldsymbol{\mu}}_p. \quad (46)$$

The described procedure is summarized in Algorithm 2. Note that the solution (46) actually corresponds to the expectation part of the product of Gaussian distributions in (45).

4 Time-driven Kernelized Movement Primitives

In many robotic tasks, such as biped locomotion (Nakanishi 2004) and striking movements (Huang et al. 2016), time plays a critical role when generating movement trajectories for a robot. We here consider a special case of KMP by taking time t as the input $\mathbf{s}$, which is aimed at learning time-driven trajectories.

4.1 A Special Treatment of Time-Driven KMP

Similarly to ProMP, we formulate a parametric trajectory comprising positions and velocities as

$$\begin{bmatrix}\boldsymbol{\xi}(t)\\ \dot{\boldsymbol{\xi}}(t)\end{bmatrix} = \boldsymbol{\Theta}(t)^{\mathsf{T}}\mathbf{w}, \quad (47)$$

where the matrix $\boldsymbol{\Theta}(t) \in \mathbb{R}^{B\mathcal{O}\times 2\mathcal{O}}$ is

$$\boldsymbol{\Theta}(t) = \begin{bmatrix}\boldsymbol{\varphi}(t) & \mathbf{0} & \cdots & \mathbf{0} & \dot{\boldsymbol{\varphi}}(t) & \mathbf{0} & \cdots & \mathbf{0}\\ \mathbf{0} & \boldsymbol{\varphi}(t) & \cdots & \mathbf{0} & \mathbf{0} & \dot{\boldsymbol{\varphi}}(t) & \cdots & \mathbf{0}\\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots\\ \mathbf{0} & \mathbf{0} & \cdots & \boldsymbol{\varphi}(t) & \mathbf{0} & \mathbf{0} & \cdots & \dot{\boldsymbol{\varphi}}(t)\end{bmatrix}. \quad (48)$$

Note that we include the first-order derivative of the parametric trajectory $\boldsymbol{\xi}(t)$ in (47), which allows us to encode the observed dynamics of the motion. Consequently, we include the derivative of basis functions as shown in (48).

In order to encapsulate the variability in demonstrations, we here model the joint probability $\mathcal{P}(t, \boldsymbol{\xi}, \dot{\boldsymbol{\xi}})$ using GMM, similarly to Section 2.1. The probabilistic reference trajectory associated with time input t_n can then be extracted by GMR as the conditional probability $\mathcal{P}(\hat{\boldsymbol{\xi}}_n, \hat{\dot{\boldsymbol{\xi}}}_n | t_n) \sim \mathcal{N}(\hat{\boldsymbol{\mu}}_n, \hat{\boldsymbol{\Sigma}}_n)$. Finally, we can derive the time-driven KMP by following the derivations presented in Section 2.2.

¶Note that, if the input $\mathbf{s}$ becomes time, then $\mathbf{A}_s^{(p)} = 1$ and $\mathbf{b}_s^{(p)} = 0$.

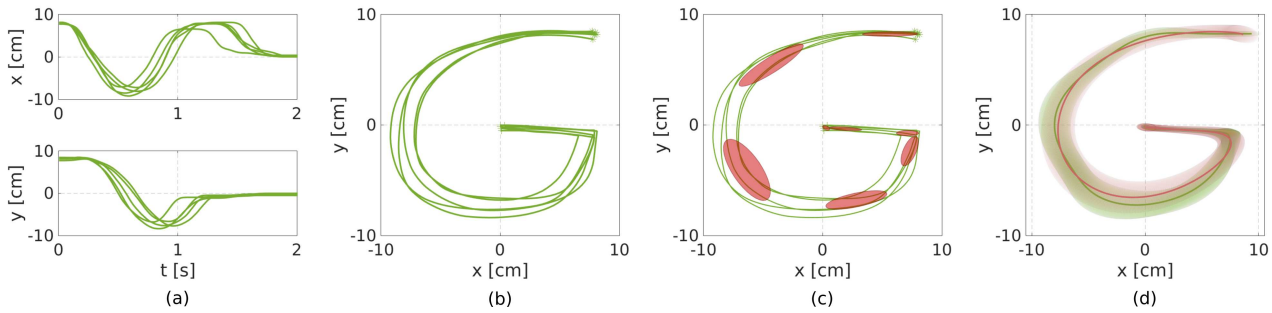


Figure 1. Learning of handwritten letters 'G'. (*a*) and (*b*) show trajectories of 'G', '$*$' and '+' in (*b*) denote the starting and ending points of the demonstrations, respectively. (*c*) depicts the estimated GMM with ellipses representing Gaussian components. (*d*) displays the reference trajectory distribution retrieved by GMR (green color) and the reproduced distribution by KMP (pink color), the curves and shaded areas, respectively, correspond to the mean and standard deviation of the trajectories.

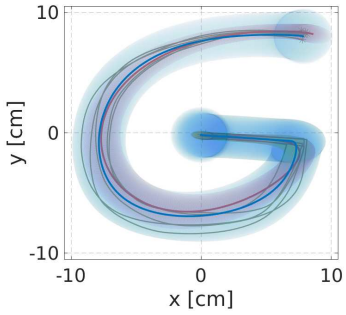


Figure 2. Reproduced trajectory distributions through KMP (pink color) and ProMP (blue color), where gray curves depict demonstrations. The pink and blue curves as well as their associated shaded areas respectively represent the means and standard deviations of the retrieved trajectories. Note that the over-estimation of variance via ProMP results from the regularized term added to $\boldsymbol{\Sigma}_w$.

It is noted that, when we calculate the kernel matrix as previously defined in (16)–(18), we here encounter four types of products $\boldsymbol{\varphi}(t_i)^{\mathsf{T}}\boldsymbol{\varphi}(t_j)$, $\boldsymbol{\varphi}(t_i)^{\mathsf{T}}\dot{\boldsymbol{\varphi}}(t_j)$, $\dot{\boldsymbol{\varphi}}(t_i)^{\mathsf{T}}\boldsymbol{\varphi}(t_j)$ and $\dot{\boldsymbol{\varphi}}(t_i)^{\mathsf{T}}\dot{\boldsymbol{\varphi}}(t_j)$. Hence, we propose to approximate $\dot{\boldsymbol{\varphi}}(t)$ as $\dot{\boldsymbol{\varphi}}(t) \approx \frac{\boldsymbol{\varphi}(t+\delta)-\boldsymbol{\varphi}(t)}{\delta}$ by using the finite difference method, where $\delta > 0$ is an extremely small constant. So, based on the definition $\boldsymbol{\varphi}(t_i)^{\mathsf{T}}\boldsymbol{\varphi}(t_j) = k(t_i, t_j)$, we can determine the kernel matrix as

$$\mathbf{k}(t_i,t_j)=\boldsymbol{\Theta}(t_i)^{\mathsf{T}}\boldsymbol{\Theta}(t_j)=\begin{bmatrix} k_{tt}(i,j)\mathbf{I}_{\mathcal{O}} & k_{td}(i,j)\mathbf{I}_{\mathcal{O}} \\ k_{dt}(i,j)\mathbf{I}_{\mathcal{O}} & k_{dd}(i,j)\mathbf{I}_{\mathcal{O}} \end{bmatrix}, \tag{49}$$

where

$$\begin{aligned} k_{tt}(i,j)&=k(t_i,t_j),\\ k_{td}(i,j)&=\frac{k(t_i,t_j+\delta)-k(t_i,t_j)}{\delta},\\ k_{dt}(i,j)&=\frac{k(t_i+\delta,t_j)-k(t_i,t_j)}{\delta},\\ k_{dd}(i,j)&=\frac{k(t_i+\delta,t_j+\delta)-k(t_i+\delta,t_j)-k(t_i,t_j+\delta)+k(t_i,t_j)}{\delta^2}. \end{aligned} \tag{50}$$

It follows that we can actually model the output variable $\boldsymbol{\xi}(t)$ and its derivative $\dot{\boldsymbol{\xi}}(t)$ in (47) using $\boldsymbol{\Theta}(t) = blockdiag(\boldsymbol{\varphi}(t), \boldsymbol{\varphi}(t), \cdots, \boldsymbol{\varphi}(t))$. In other words, the derivative of basis functions is not used. However, this treatment requires a higher dimensional $\boldsymbol{\Theta}(t)$, (i.e., $2B\mathcal{O} \times 2\mathcal{O}$) and thus leads to a higher dimensional $\mathbf{w} \in \mathbb{R}^{2B\mathcal{O}}$. In contrast, if both basis functions and their derivatives (as defined in (48)) are employed, we can obtain a compact representation which essentially corresponds to a lower dimensional $\mathbf{w} \in \mathbb{R}^{B\mathcal{O}}$.

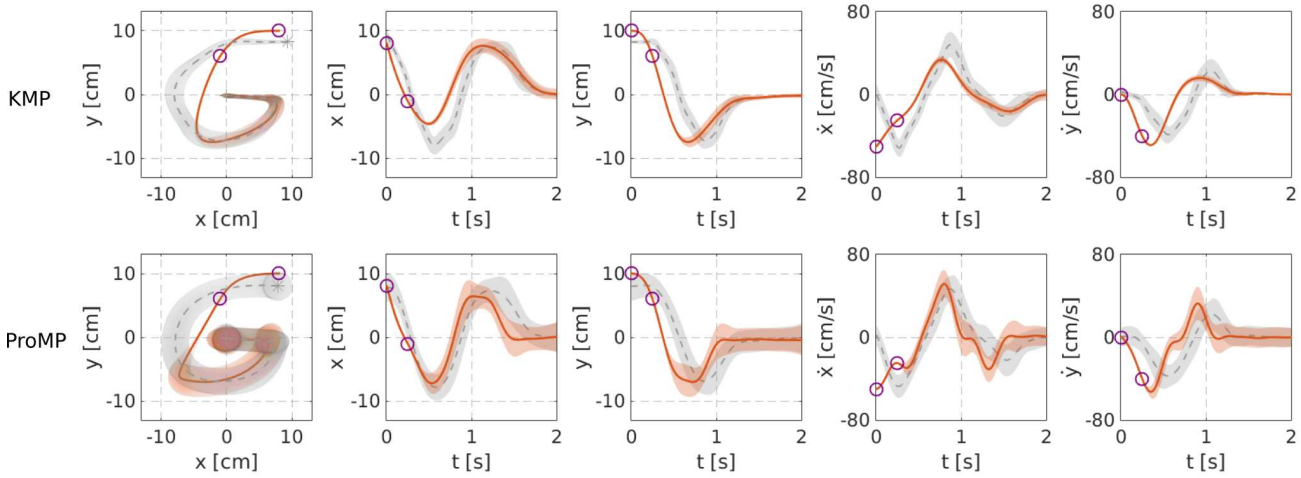
While the derivation presented in this section applies for a time-driven case, it cannot be easily generalized to the case of high-dimensional $\mathbf{s}$. Unlike a straightforward approximation of $\dot{\boldsymbol{\varphi}}(t)$ by using the finite difference method, for the high-dimensional input $\mathbf{s}$ it is a non-trivial problem to estimate $\dot{\boldsymbol{\varphi}}(\mathbf{s}) = \frac{\partial \boldsymbol{\varphi}(\mathbf{s})}{\partial \mathbf{s}} \frac{\partial \mathbf{s}}{\partial t}$ unless we have an additional model which can reflect the dynamics between time t and the input $\mathbf{s}$. Due to the difficulty of estimating $\dot{\boldsymbol{\varphi}}(\mathbf{s})$, an alternative way to encode $[\boldsymbol{\xi}^{\mathsf{T}}(\mathbf{s})\ \dot{\boldsymbol{\xi}}^{\mathsf{T}}(\mathbf{s})]^{\mathsf{T}}$ with high-dimensional input $\mathbf{s}$ is to use (2) with an extended matrix $\boldsymbol{\Theta}(\mathbf{s}) \in \mathbb{R}^{2B\mathcal{O}\times 2\mathcal{O}}$, i.e., $\boldsymbol{\Theta}(\mathbf{s}) = blockdiag(\boldsymbol{\varphi}(\mathbf{s}), \boldsymbol{\varphi}(\mathbf{s}), \cdots, \boldsymbol{\varphi}(\mathbf{s}))$.

4.2 Time-scale Modulation of Time-driven KMP

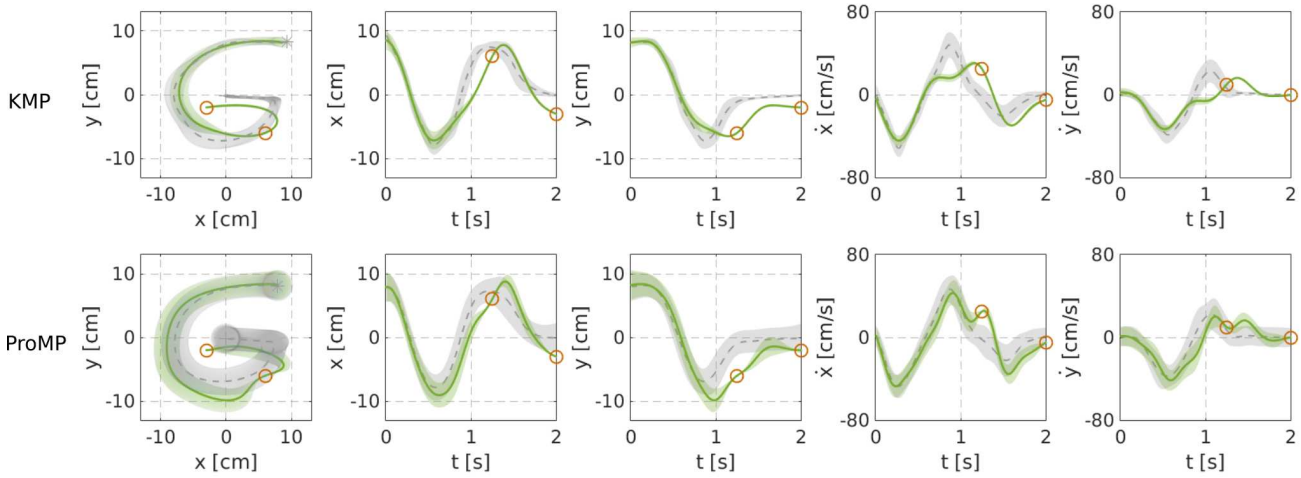
In the context of time-driven trajectories, new tasks may demand to speed up or slow down the robot movement, and hence the trajectory modulation on the time-scale is required. Let us denote the movement duration of demonstrations and the time length of the corresponding reference trajectory as t_N. To generate adapted trajectories with new durations t_D, we define a monotonic increasing function $\tau : [0, t_D] \mapsto [0, t_N]$, which is a transformation of time. This straightforward solution implies that for any query $t^* \in [0, t_D]$, we use $\tau(t^*)$ as the input for the prediction through KMP, and thus trajectories can be modulated as faster or slower (see also Ijspeert et al. (2013); Paraschos et al. (2013) for the modulations in time-scale, where the time modulation is called the phase transformation.).

5 Evaluations of the Approach

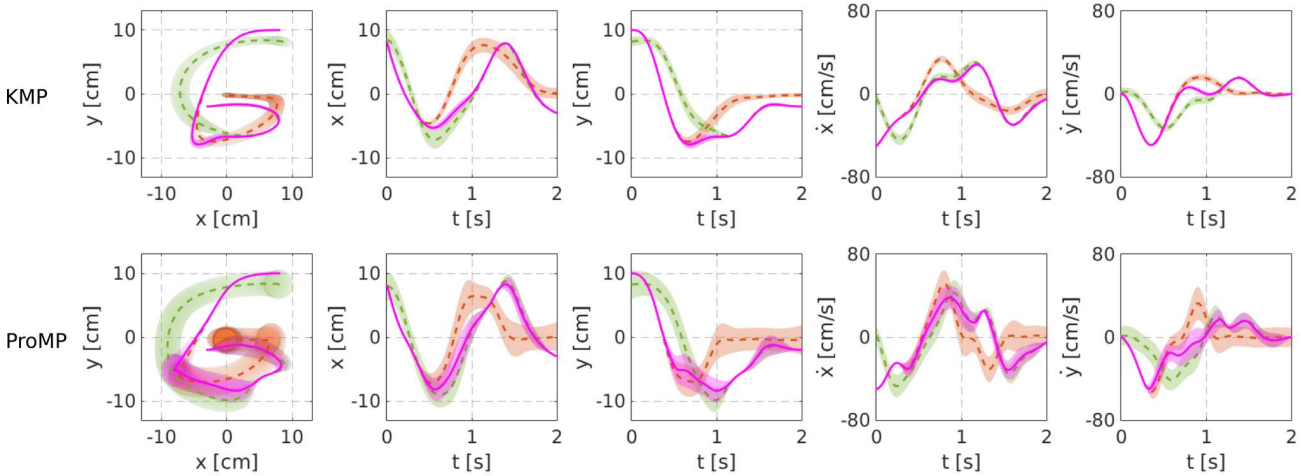
In this section, several examples are used to evaluate KMP. We first consider the adaptation of trajectories with via-points/end-points as well as the mixture of multiple trajectories (Section 5.1), where comparisons with ProMP are shown. Then, we evaluate the extrapolation capabilities of local-KMPs (Section 5.2). Subsequently, we validate the approach in two different scenarios using real robots. First, we study a novel application of robot motion



(a) Evaluation 1: trajectory modulations with one start-point and one via-point.



(b) Evaluation 2: trajectory modulations with one via-point and one end-point.



(c) Evaluation 3: Superposition of two probabilistic reference trajectories.

Figure 3. Different cases of trajectory modulation using KMP and ProMP. *(a)–(b)* show trajectories (red and green curves) that are adapted to go through different desired positions and velocities (depicted by circles). The gray curves represent the mean of probabilistic reference trajectories for KMP and ProMP, while the shaded areas depict the standard deviation. *(c)* shows the superposition of various reference trajectories, where the dashed red and green curves correspond to the adapted trajectories in *(a)* and *(b)*, respectively. The resulting trajectory is displayed in solid pink curve.

adaptation by adding via-points according to sensed forces at the end-effector of the robot (Section 5.3). Second, we focus on a human-robot collaboration scenario, namely, the collaborative hand task, where a 6-dimensional input is considered in the learning and adaptation problems (Section 5.4).

5.1 Trajectory Modulation and Superposition

We first evaluate our approach using five trajectories of the handwritten letter 'G'[‖], as shown in Figure 1(*a*)-(*b*). These demonstrations are encoded by GMM with input t and output $\boldsymbol{\xi}(t)$ being the 2-D position $[x(t)\, y(t)]^{\mathsf{T}}$. Subsequently, a probabilistic reference trajectory is retrieved through GMR, as depicted in Figure 1(*c*)–(*d*), where the position values from the reference trajectory are shown. This probabilistic reference trajectory along with the input is used to initialize KMP as described in Section 4.1, which uses a Gaussian kernel $k(t_i, t_j) = exp(-\ell(t_i - t_j)^2)$ with hyperparameter $\ell > 0$. The relevant parameters for KMP are set as $\ell = 4$, $\lambda = 0.5$ and $\lambda_c = 60$. The reproduction of KMP is provided in Figure 1(*d*), showing that the resulting distribution (pink color) resembles the GMR-based retrieved distribution (green color).

For comparison purposes, ProMP is evaluated as well, where 21 Gaussian basis functions chosen empirically are used. For each demonstration, we employ the regularized least squares method to estimate the weights $\mathbf{w} \in \mathbb{R}^{42}$ of the corresponding basis functions. Subsequently, the probability distribution $\mathcal{P}(\boldsymbol{\mu}_w, \boldsymbol{\Sigma}_w)$ that is computed through maximum likelihood estimation (Paraschos et al. 2015) is used to initialize ProMP. Due to the number of demonstrations being significantly lower than the dimension of $\mathbf{w}$, a diagonal regularized term is added to $\boldsymbol{\Sigma}_w$ to avoid singular estimations as well as singularity of matrix inverse operation in trajectory adaptations[**]. For the sake of clear observation, the reproduced trajectories through KMP and ProMP are displayed in Figure 2.

Figure 3 displays different trajectory modulation cases using KMP and ProMP. We test not only cases in which new requirements arise in the form of via-points and start-points/end-points, but also the scenario of mixing different reference trajectories. It can be observed from Figure 3(*a*)–(*b*) that both KMP and ProMP successfully generate trajectories that fulfill the new requirements, however, KMP generates smoother trajectories. A quantitative comparison over smoothness is summarized in Table 2, where the position smoothness c_p is defined as $c_p = \frac{1}{N}\sum_{t=1}^{N-1}(\boldsymbol{\xi}_{t+1} - \boldsymbol{\xi}_t)^{\mathsf{T}}(\boldsymbol{\xi}_{t+1} - \boldsymbol{\xi}_t)$, and velocity smoothness c_v is computed similarly. For the case of trajectory superposition in Figure 3(*c*), we consider the adapted trajectories in Figure 3(*a*) and (*b*) as candidate reference trajectories and assign them with the priorities $\gamma_{t,1} = \exp(-t)$ and $\gamma_{t,2} = 1 - \exp(-t)$, respectively. Note that $\gamma_{t,1}$ and $\gamma_{t,2}$ correspond to monotonically decreasing and increasing functions, respectively. As depicted in Figure 3(*c*), the mixed trajectory (solid pink curve) indeed switches from the first to the second reference trajectory.

Despite KMP and ProMP can adapt trajectories towards various desired points, one key difference between them lies on the determination of basis functions. In contrast to ProMP that requires explicit basis functions, KMP is a non-parametric method that does not depend on explicit basis functions, given that the probabilistic reference trajectory is provided beforehand. This difference proves to be substantially crucial for tasks where the robot actions are driven by a high-dimensional input. We will show this effect in the collaborative hand task (Section 5.4) which is associated with a 6-D input, where the implementation of ProMP becomes difficult since a large number of basis functions need to be defined.

Table 2. Cost Values of Trajectory Smoothness

	KMP Evaluation 1	ProMP Evaluation 1	KMP Evaluation 2	ProMP Evaluation 2
c_p	0.019	0.025	0.019	0.026
c_v	0.431	0.976	0.484	0.823

5.2 Extrapolation with Local-KMPs

We evaluate the extrapolation capabilities of local-KMPs in an application with a new set of desired points (i.e., start-, via- and end-points) lying far away from the area covered by the original demonstrations, in contrast to the experiment reported in Section 5.1. Note that the original ProMP (Paraschos et al. (2013)) has a limited extrapolation capability (see Havoutis and Calinon (2017) for a discussion). Thus, we only evaluate our approach here.

We study a collaborative object transportation task, where the robot assists a human to carry an object from a starting point to an ending location. Five demonstrations in the robot base frame are used for the training of local-KMPs (see Figure 4(*a*)). We consider time t as the input, and the 3-D Cartesian position $[x(t)\, y(t)\, z(t)]^{\mathsf{T}}$ of the robot end-effector as the output $\boldsymbol{\xi}(t)$. For the implementation of local-KMPs, we define two frames located at the initial and the final locations of the transportation trajectories (as depicted in Figure 4(*b*)–(*c*)), similarly to Rozo et al. (2015), which are then used to extract the local motion patterns.

We consider two extrapolation tests, where the starting and ending locations are different from the demonstrated ones. In the first test, we study the transportation from $\mathbf{p}_s = [-0.2\ 0.2\ 0.2]^{\mathsf{T}}$ to $\mathbf{p}_e = [-0.15\ 0.8\ 0.1]^{\mathsf{T}}$. In the second test, we evaluate the extrapolation with $\mathbf{p}_s = [0.2\ \ -0.3\ 0.1]^{\mathsf{T}}$ and $\mathbf{p}_e = [0.25\ 0.5\ 0.05]^{\mathsf{T}}$. Note that all locations are described with respect to the robot base frame. In addition to the desired starting and ending locations in the transportation task, we also introduce additional position constraints which require the robot passing through two via-points (plotted by circles in Figure 5). The extrapolation of local-KMPs for these new situations is achieved according to Algorithm 2, where the Gaussian kernel is used. For each test, the local frames are set as $\mathbf{A}^{(1)} = \mathbf{A}^{(2)} = \mathbf{I}_3$, $\mathbf{b}^{(1)} = \mathbf{p}_s$ and $\mathbf{b}^{(2)} = \mathbf{p}_e$. The related KMP parameters are $\ell = 0.5$ and $\lambda = \lambda_c = 10$. Figure 5 shows that local-KMPs successfully extrapolate to new frame locations and lead the robot to go through various new desired points while maintaining the shape of the demonstrated trajectories.

[‖] These trajectories are obtained from Calinon and Lee (2017).

[**] The importance of regularization is discussed in Paraschos et al. (2018).

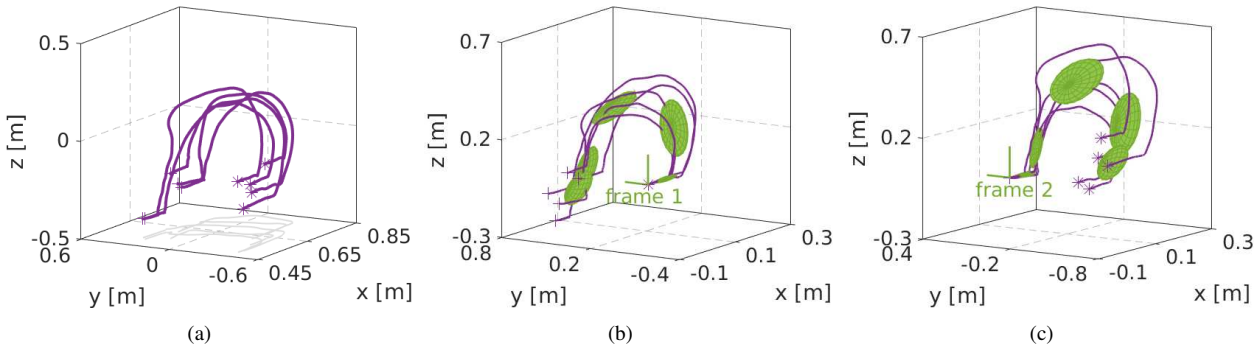


Figure 4. Demonstrations of the transportation task as well as GMM modeling of local trajectories. (*a*) shows the demonstrated trajectories (plotted by purple curves), where gray curves correspond to the projection of demonstrated trajectories into the x–y plane. '$*$' and '+' denote the starting and ending points of trajectories, respectively. (*b*)-(*c*) depict GMM modeling of local trajectories, where local trajectories are obtained by projecting demonstrations into two local frames, respectively.

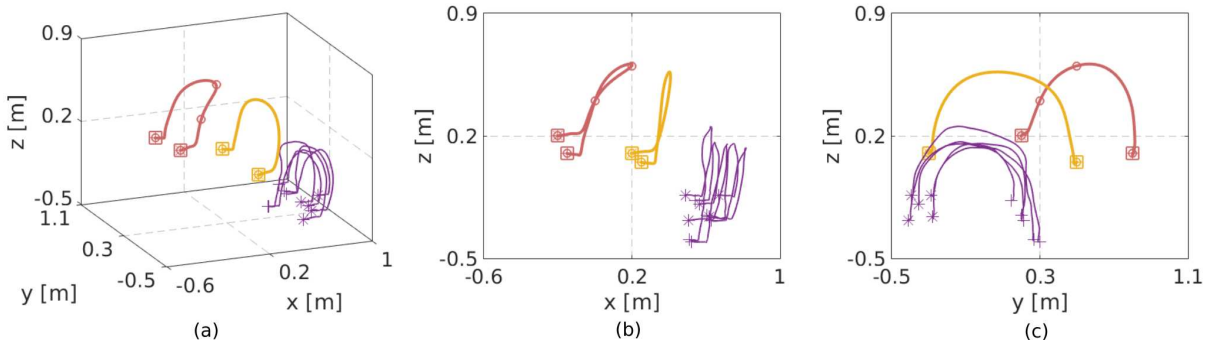


Figure 5. Extrapolation evaluations of local-KMPs for new starting and ending locations in the transportation task. *(a)* shows various trajectories in 3-D view, while *(b)* and *(c)* display projections of trajectories from *(a)* into $x-z$ plane and $y-z$ plane, respectively. The purple curves represent demonstrated trajectories, while the red and yellow trajectories show the extrapolation cases. Circles represent desired points describing additional task requirements. Squares denote desired starting and ending locations of the transportation task.

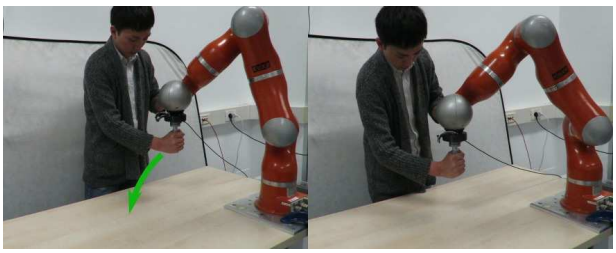


Figure 6. Kinesthetic teaching of the reaching task on the KUKA robot, where demonstrations comprising time and end-effector Cartesian position are collected. The green arrow shows the motion direction of the robot.

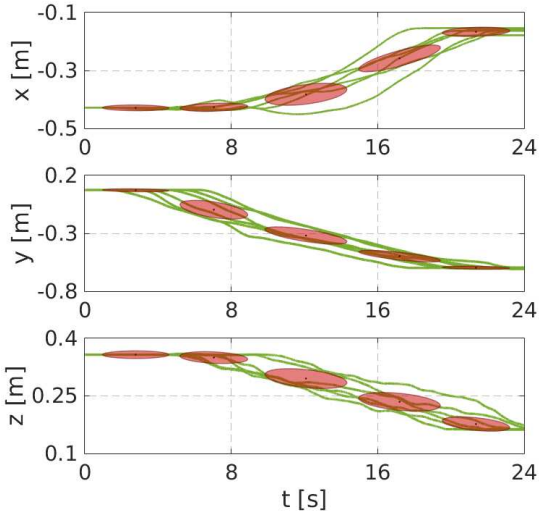


Figure 7. GMM modeling of demonstrations for the force-based adaptation task, where the green curves represent demonstrated trajectories and ellipses depict Gaussian components.

Note that the environment might drastically change from demonstrations to final execution, so the capability of modulating the demonstrated trajectories to go through new points is important in many applications. In this sense, local-KMPs prove superior to other local-frame approaches such as those exploited in Rozo et al. (2015); Calinon (2016), which do not consider trajectory modulation.

5.3 Force-based Trajectory Adaptation

Through kinesthetic teaching, humans are able to provide the robot with initial feasible trajectories. However, this procedure does not account for unpredicted situations. For instance, when the robot is moving towards a target, undesired circumstances such as obstacles occupying the robot workspace might appear, which requires the robot to avoid possible collisions. Since humans have reliable reactions over dynamic environments, we here propose to

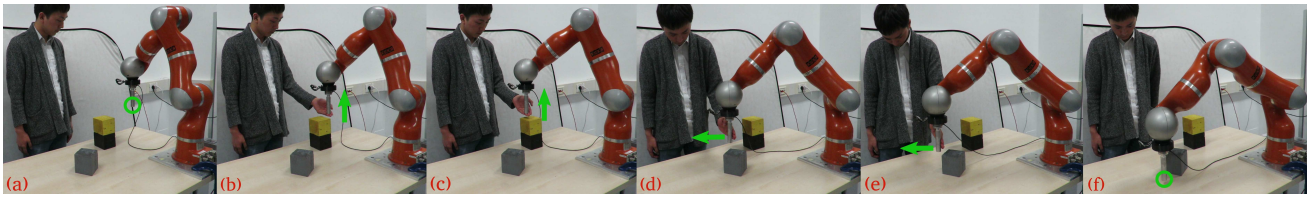


Figure 8. Snapshots of the force-based trajectory adaptations, where the force exerted by the human is used to determine the via-points for the robot, which ensures collision avoidance. (a) and (f) correspond to the initial and final states of the robot, where circles depict the initial and final positions, respectively. Figures (b)–(e) show human interactions with the green arrows depicting the directions of corrective force.

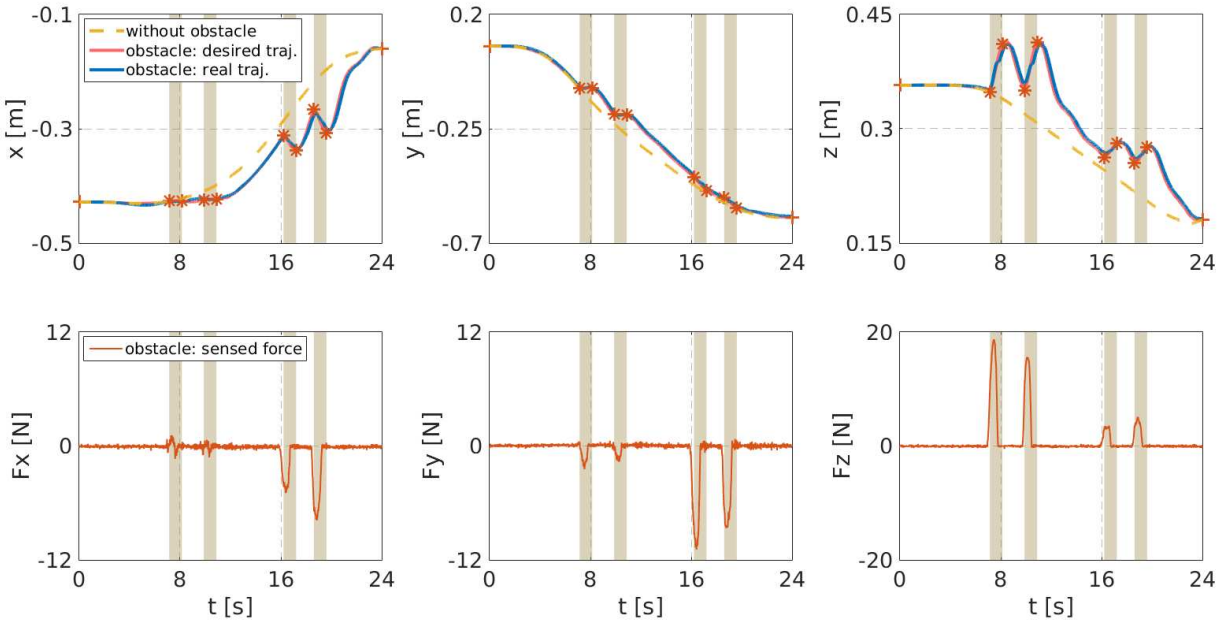


Figure 9. *Top row*: the desired trajectory (generated by KMP) and the real robot trajectory, where '$*$' represents the force-based desired points and '+' corresponds to the initial and final locations for the robot. For comparison, we also provide the desired trajectory predicted by KMP without obstacles (i.e., without human intervention). The shaded areas show the regulation durations for various human interventions. *Bottom row*: the force measured at the end-effector of the KUKA robot.

use the human supervision to adapt the robot trajectory when the environment changes. In particular, we use a force sensor installed at the end-effector of the robot in order to measure corrective forces exerted by the human.

We treat the force-based adaptation problem under the KMP framework by defining new via-points as a function of the sensed forces. Whenever the robot is about to collide with the obstacle, the user interacts physically with the end-effector and applies a corrective force. This force is used to determine a desired via-point which the robot needs to pass through in order to avoid the obstacle. By updating the reference database using this obtained via-point through (34), KMP can generate an adapted trajectory that fulfills the via-point constraint.

For the human interaction at time t, given the robot Cartesian position $\mathbf{p}_t$ and the sensed force $\mathbf{F}_t$, the first desired datapoint is defined as: $\bar{t}_1 = t + \delta_t$ and $\bar{\mathbf{p}}_1 = \mathbf{p}_t + \mathbf{K}_f \mathbf{F}_t$, where $\delta_t > 0$ controls the regulation time and $\mathbf{K}_f > 0$ determines the adaptation proportion for the robot trajectory. In order to avoid undesired trajectory modulations caused by the force sensor noise, we introduce a force threshold F_{thre} and add the new force-based via-point to the reference trajectory only when $||\mathbf{F}_t|| > F_{thre}$. Note that the adapted trajectory might be far away from the previous planned trajectory due to the new via-point, we hence consider adding $\mathbf{p}_t$ as the second desired point so as to ensure a smooth trajectory for the robot. Doing so, for each interaction, we define the second desired point as $\bar{t}_2 = t$ and $\bar{\mathbf{p}}_2 = \mathbf{p}_t$.

In order to evaluate the adaptation capability of KMP, we consider a reaching task where unpredicted obstacles will intersect the robot movement path. First, we collect six demonstrations (as depicted in Figure 6) comprising time input t and output $\boldsymbol{\xi}(t)$ being the 3-D Cartesian position $[x(t)\, y(t)\, z(t)]^{\mathsf{T}}$. Note that obstacles are not placed in the training phase. The collected data is fitted using GMM (plotted in Figure 7) so as to retrieve a reference database, which is subsequently used to initialize KMP. Then, during the evaluation phase, two obstacles whose locations intersect the robot path are placed on the table, as shown in Figure 8. In addition to the via-points that will be added through physical interaction, we add the initial and target locations for the robot as desired points beforehand, where the initial location corresponds to the robot position before starting to move. The relevant parameters are $\mathbf{K}_f$=$0.006\mathbf{I}_3$, $\delta_t = 1s$, $F_{thre} = 10N$, $\ell = 0.15$ and $\lambda = \lambda_c = 0.3$.

The trajectory that is generated by KMP according to various desired points as well as the real robot trajectory

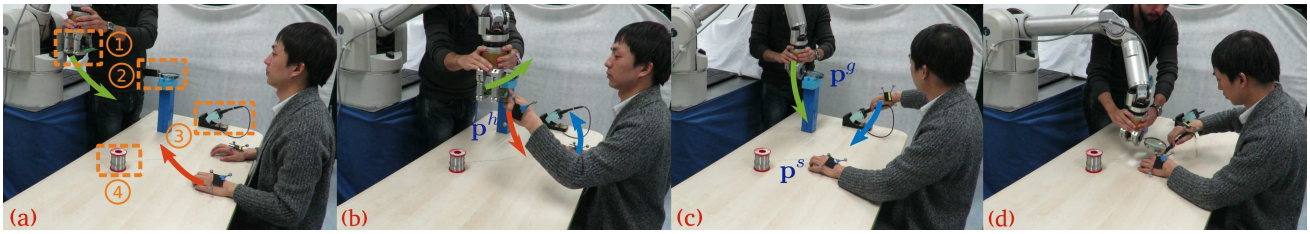
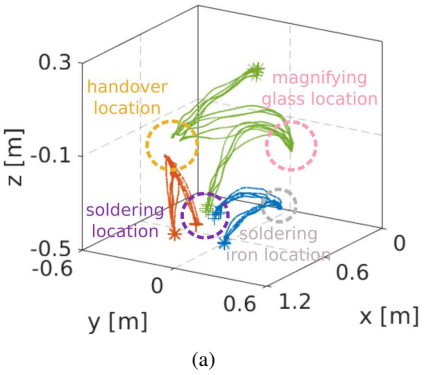


Figure 10. The collaborative hand task in the soldering environment with the Barrett WAM robot. *(a)* shows the initial states of the user hands and the robot end-effector (the collaborative hand in this experiment). ①–④ separately correspond to the circuit board (held by the robot), magnifying glass, soldering iron and solder. *(b)* corresponds to the handover of the circuit board. *(c)* shows the robot grasping of the magnifying glass. *(d)* depicts the final scenario of the soldering task using both of the user hands and the robot end-effector. Red, blue and green arrows depict the movement directions of the user left hand, right hand and the robot end-effector, respectively.



(a)

Figure 11. Demonstrations for the collaborative hand task, where the red and blue curves respectively correspond to the user left and right hands, while the green curves represent the demonstrated trajectories for the robot. The '∗' and '+' mark the starting and ending points of various trajectories, respectively.

are depicted in Figure 9. We can observe that for each obstacle the robot trajectory is adapted twice. In the first two adaptations (around $8s$ and $11s$), the corrective force is dominant along the z direction, while in the last two adaptations (around $17s$ and $20s$), the force has a larger component along the x and y directions. For all cases, KMP successfully adapts the end-effector trajectory according to the measured forces.

Note that, even without human interaction, the proposed scheme can also help the robot replan its trajectory when it touches the obstacles, where the collision force takes the role of the human correction and guides the robot to move away from the obstacles. Thus, with KMP the robot is capable of autonomously adapting its trajectory through low-impact collisions, whose tolerated force can be regulated using F_{thre}. Supplementary material includes a video of experiments using the human corrective force and the obstacle collision force.

5.4 Collaborative Hand Task

So far the reported experiments have shown the performances of KMP by learning various time-driven trajectories. We now consider a different task which requires a 6-D input, in particular a robot-assisted soldering scenario. As shown in Figure 10, the task proceeds as follows: *(1)* the robot needs to hand over a circuit board to the user at the *handover location* $\mathbf{p}^h$ (Figure 10*(b)*), where the left hand of the user is used. *(2)* the user moves his left hand to place the circuit board at the *soldering location* $\mathbf{p}^s$ and simultaneously moves his right hand towards the soldering iron and then grasps it. Meanwhile, the robot is required to move towards the magnifying glass and grasp it at the *magnifying glass location* $\mathbf{p}^g$ (Figure 10*(c)*). *(3)* the user moves his right hand to the soldering location so as to repair the circuit board. Meanwhile, the robot, holding the magnifying glass, moves towards the soldering place in order to allow the user to take a better look at the small components of the board (Figure 10*(d)*).

Let us denote $\mathbf{p}^{\mathcal{H}_l}$, $\mathbf{p}^{\mathcal{H}_r}$ and $\mathbf{p}^{\mathcal{R}}$ as positions of the user left hand, right hand and robot end-effector (i.e., the "collaborative hand"), respectively. Since the robot is required to react properly according to the user hand positions, we formulate the collaborative hand task as the prediction of the robot end-effector position according to the user hand positions. In other words, in the prediction problem we consider $\mathbf{s} = \{\mathbf{p}^{\mathcal{H}_l}, \mathbf{p}^{\mathcal{H}_r}\}$ as the input (6-D) and $\boldsymbol{\xi}(\mathbf{s}) = \mathbf{p}^{\mathcal{R}}$ as the output (3-D) .

Following the procedure illustrated in Figure 10, we collect five demonstrations comprising $\{\mathbf{p}^{\mathcal{H}_l}, \mathbf{p}^{\mathcal{H}_r}, \mathbf{p}^{\mathcal{R}}\}$ for training KMP, as shown in Figure 11. Note that the teacher only gets involved in the training phase. We fit the collected data using GMM, and subsequently extract a probabilistic reference trajectory using GMR, where the input for the probabilistic reference trajectory is sampled from the marginal probability distribution $\mathcal{P}(\mathbf{s})$, since in this scenario the exact input is unknown (unlike time t in previous experiments). The Gaussian kernel is also employed in KMP, whose hyperparameters are set to $\ell = 0.5$ and $\lambda = \lambda_c = 2$.

Two evaluations are carried out to evaluate KMP in this scenario. First, we employ the learned reference database without adaptation so as to verify the reproduction ability of KMP, as shown in Figure 12 (*top row*). The user left- and right-hand trajectories as well as the real robot trajectory, depicted in Figure 12 (*top row*), are plotted in Figure 13 (dotted curves), where the desired trajectory for robot end-effector is generated by KMP. We can observe that KMP maintains the shape of the demonstrated trajectories for the robot while accomplishing the soldering task. Second, we evaluate the adaptation capability of KMP by adjusting the handover location $\mathbf{p}^h$, the magnifying glass location $\mathbf{p}^g$ as well as the soldering location $\mathbf{p}^s$, as illustrated in Figure 12 (*bottom row*). Note that these new locations are unseen in the

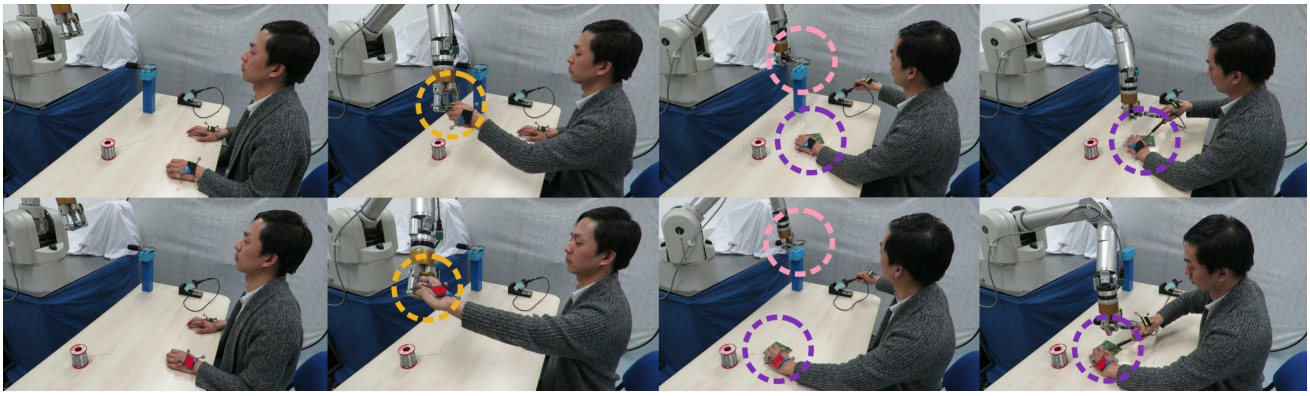
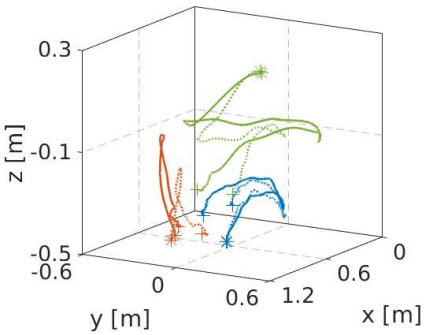


Figure 12. Snapshots of reproduction and adaptation using KMP. *Top row* shows the reproduction case using the learned reference database without adaptation. *Bottom row* displays the adaptation case using the new reference database which is updated using three new desired points: new handover, magnifying glass and soldering locations depicted as dashed circles (notice the difference with respect to the top row).



(a)

Figure 13. The reproduction (dotted curves) and adaptation (solid curves) capabilities of KMP in the collaborative hand task, where the user left-hand and right-hand trajectories (red and blue curves) are used to retrieve the robot end-effector trajectory (green curves).

demonstrations, thus we consider them as new via-point/end-point constraints within the KMP framework.

To take the handover as an example, we can define a via-point (associated with input) as $\{\bar{\mathbf{p}}_1^{\mathcal{H}_l}, \bar{\mathbf{p}}_1^{\mathcal{H}_r}, \bar{\mathbf{p}}_1^{\mathcal{R}}\}$, where $\bar{\mathbf{p}}_1^{\mathcal{H}_l} = \mathbf{p}^h$, $\bar{\mathbf{p}}_1^{\mathcal{H}_r} = \mathbf{p}_{ini}^{\mathcal{H}_r}$ and $\bar{\mathbf{p}}_1^{\mathcal{R}} = \mathbf{p}^h$, which implies that the robot should reach the new handover location $\mathbf{p}^h$ when the user left hand arrives at $\mathbf{p}^h$ and the user right hand stays at its initial position $\mathbf{p}_{ini}^{\mathcal{H}_r}$. Similarly, we can define additional via- and end-points to ensure that the robot grasps the magnifying glass at a new location $\mathbf{p}^g$ and assists the user at a new location $\mathbf{p}^s$. Thus, two via-points and one end-point are used to update the original reference database according to (34) so as to address the three adaptation situations. Figure 13 shows the adaptations of the robot trajectory (green solid curve) in accordance with the user hand trajectories (red and blue solid curves). It can be seen that the robot trajectory is indeed modulated towards the new handover, magnifying glass and soldering locations, showing the capability of KMP to adapt trajectories associated with high-dimensional inputs.

It is worth pointing out that the entire soldering task is accomplished by a single KMP without any trajectory segmentation for different subtasks, thus allowing for a straightforward learning of several sequential subtasks. Moreover, KMP makes the adaptation of learned skills associated with high-dimensional inputs feasible. Also, KMP is driven by the user hand positions, which allows for slower/faster hand movements since the prediction of KMP does not depend on time, hence alleviating the typical problem of time-alignment (Amor et al. 2014) and phase estimation (Maeda et al. 2017) in human-robot collaborations. For details on the collaborative hand experiments, please refer to the video in the supplementary material.

6 Related Work

In light of the reliable temporal and spatial generalization, DMP (Ijspeert et al. 2013) has achieved remarkable success in a vast range of applications. In addition, many variants of DMP have been developed for specific circumstances, such as stylistic DMP (Matsubara et al. 2010), task-parameterized DMP (Pervez and Lee 2017), combined DMP (Pastor et al. 2009) and cooperative DMP (Gams et al. 2014). However, due to the spring-damper dynamics, DMP converges to the target position with zero velocity, which prevents its application to cases with velocity requirements (e.g., the striking/batting movement). Besides, DMP does not provide a straightforward way to incorporate desired via-points.

By exploiting the properties of Gaussian distributions, ProMP (Paraschos et al. 2013) allows for trajectory adaptations with via-points and end-points simultaneously. The similarities between DMP and ProMP lie on the fact that both methods need the explicit definition of basis functions and are aimed at learning time-driven trajectories. As a consequence, when we encounter trajectories with high-dimensional inputs (e.g., human hand position and posture in human-robot collaboration scenarios), the selection of basis functions in DMP and ProMP becomes difficult and thus undesired.

Note that DMP (Amor et al. 2014) and ProMP (Maeda et al. 2017; Ewerton et al. 2015) were applied in human-robot collaboration scenarios, where human and robot trajectories were encoded with time. When human actions are adjusted with time, various algorithms are provided to predict the corresponding robot motions with time. Since human trajectories could be different (e.g., moving

with faster/slower speeds) from the demonstrated ones, additional time alignment or phase estimation is required in order to synchronize human and robots. In contrast, in our collaborative hand experiment, we explicitly take the positions of the user's two hands as the input, and, subsequently, predict robot trajectories as the output, which did not require any time-alignment process.

Differing from DMP and ProMP, GMM/GMR based learning algorithms (Muhlig et al. 2009; Calinon et al. 2007) have been proven effective in encoding demonstrations with high-dimensional inputs. However, the large number of variables arising in GMM makes the re-optimization of GMM expensive, which therefore prevents its use in unstructured environments where robot adaptation capabilities are imperative.

KMP provides several advantages compared to the aforementioned works. Unlike GMM/GMR, KMP is capable of adapting trajectories towards various via-points/end-points without the optimization of high-dimensional hyperparameters. Unlike DMP and ProMP, KMP alleviates the need of explicit basis functions due to its kernel treatment, and thus can be easily implemented for problems with high-dimensional inputs and outputs.

It is noted that the training of DMP only needs a single demonstration, while ProMP, GMM and KMP require a set of trajectories. In contrast to the learning of a single demonstration, the exploitation of multiple demonstrations makes the extraction of probabilistic properties of human skills possible. In this context, demonstrations have been exploited using the covariance-weighted strategy, as in trajectory-GMM (Calinon 2016), linear quadratic regulators (LQR) (Rozo et al. 2015), movement similarity criterion (Muhlig et al. 2009) and demonstration-guided trajectory optimization (Osa et al. 2017). Note that the mean minimization subproblem as formulated in (10) also uses the covariance to weigh the cost, sharing the same spirit of the aforementioned results.

Similarly to our approach, information theory has also been exploited in different robot learning techniques. As an effective way to measure the distance between two probabilistic distributions, KL-divergence was exploited in policy search (Peters et al. 2010; Kahn et al. 2017), trajectory optimization (Levine and Abbeel 2014) and imitation learning (Englert et al. 2013). In Englert et al. (2013) KL-divergence was used to measure the difference between the distributions of demonstrations and the predicted robot trajectories (obtained from a control policy and a Gaussian process forward model), and subsequently the probabilistic inference for learning control (Deisenroth and Rasmussen 2011) was employed to iteratively minimize the KL-divergence so as to find the optimal policy parameters. It is noted that this KL-divergence formulation makes the derivations of analytical solution intractable. In this article, we formulate the trajectory matching problem as (5), which allows us to separate the mean and covariance subproblems and derive closed-form solutions for them separately.

Other related work is Gaussian process based probabilistic inference (Dong et al. 2016; Rana et al. 2017). In Rana et al. (2017), demonstrated trajectories were encoded by Gaussian process as a prior, and subsequently additional constraints (e.g., obstacle avoidance) were represented in the form of likelihood. Through maximizing the posterior probability, optimal trajectories were estimated. In comparison with their work, we study imitation learning from an information theory perspective. Specifically, we exploit the covariance (i.e., variability) of demonstrations while in Dong et al. (2016); Rana et al. (2017) the covariance relies on the input distribution and thus the consistent or important features of demonstrations are not exploited.

7 Discussion

While both KMP and ProMP (Paraschos et al. 2013) learn the probabilistic properties of demonstrations, we here discuss their similarities and possible shortcomings in detail. For the KMP, imitation learning is formulated as an optimization problem (Section 2.2.1), where the optimal distribution $\mathcal{N}(\boldsymbol{\mu}_w^*, \boldsymbol{\Sigma}_w^*)$ of $\mathbf{w}$ is derived by minimizing the information-loss between the parametric trajectory and the demonstrations. Specifically, the mean minimization subproblem (10) can be viewed as the problem of maximizing the posterior $\prod_{n=1}^{N} \mathcal{P}(\boldsymbol{\Theta}(\mathbf{s}_n)^{\mathsf{T}}\boldsymbol{\mu}_w|\hat{\boldsymbol{\mu}}_n, \hat{\boldsymbol{\Sigma}}_n)$. In contrast, ProMP formulates the problem of imitation learning as an estimation of the probability distribution of movement pattern $\mathbf{w}$ (i.e., $\mathbf{w} \sim \mathcal{N}(\boldsymbol{\mu}_w, \boldsymbol{\Sigma}_w)$), which is essentially equivalent to the maximization of the likelihood $\prod_{h=1}^{H}\prod_{n=1}^{N} \mathcal{P}(\boldsymbol{\xi}_{n,h}|\boldsymbol{\Theta}(\mathbf{s}_n)^{\mathsf{T}}\boldsymbol{\mu}_w, \boldsymbol{\Theta}(\mathbf{s}_n)^{\mathsf{T}}\boldsymbol{\Sigma}_w\boldsymbol{\Theta}(\mathbf{s}_n))$. To solve this maximization problem, the regularized least-squares is first used for each demonstration so as to estimate its corresponding movement pattern vector (Paraschos et al. 2015), where basis functions are used to fit these demonstrations. Subsequently, using the movement patterns extracted from demonstrations, the distribution $\mathcal{P}(\mathbf{w})$ is determined by using the maximum likelihood estimation.

A direct problem in ProMP is the estimation of $\mathcal{P}(\mathbf{w})$. If the dimension of $\mathbf{w}$ (i.e., $B\mathcal{O}$) is too high compared to the number of demonstrations H, a singular covariance $\boldsymbol{\Sigma}_w$ might appear. For this reason, learning movements with ProMP typically requires a high number of demonstrations. In contrast, KMP needs a probabilistic reference trajectory, which is derived from the joint probability distribution of $\{\mathbf{s}, \boldsymbol{\xi}\}$ that is typically characterized by a lower dimensionality (i.e., $\mathcal{I}+\mathcal{O}$). Another problem in ProMP comes up with demonstrations with high dimensional input $\mathbf{s}$, where the number of basis functions increases with the dimension of the input, which is the typical curse of dimensionality (see also the discussion on the disadvantages of fixed basis functions in Bishop (2006)). In contrast, KMP is combined with a kernel function, alleviating the need for basis functions, while inheriting all the potential and expressiveness of kernel-based methods.

In this paper we mainly focus on robot skill learning at the level of trajectory generation. From an implementation perspective, we could employ various optimal controllers, such as LQR and model predictive control (MPC), to drive robots to execute the planned trajectories from KMP. For instance, as KMP predicts mean and covariance of trajectory points simultaneously, we may employ the minimal intervention control scheme in Medina et al. (2012) and set the mean and the inverse of covariance as the desired reference trajectory and the weight matrix for tracking errors in LQR, respectively. In this case, the

tracking performance is governed by trajectory covariances. Specifically, small covariances require small tracking errors while large covariances allow for large errors.

There are several possible extensions for KMP. First, similarly to most regression algorithms, the computation complexity of KMP increases with the size of training data (i.e., the reference database in our case). One possible solution could be the use of partial training data so as to build a sparse model (Bishop 2006). Second, even though we have shown the capability of KMP on trajectory adaptation, the choice of desired points is rather empirical. For more complicated situations where we have no (or minor) prior information, the search of optimal desired points could be useful. To address this problem, RL algorithms could be employed to find appropriate new via-points that fulfill the relevant task requirements. Finally, the choice of kernel in KMP is crucial. In our evaluations, the frequently used Gaussian kernel was applied. Generally, it is non-trivial to determine the optimal kernel beforehand. One typical solution is to choose the kernel according to task requirements at hand (in our case the kernel should depend on the nature of demonstrations). Since various kernels have been developed and many insights are provided (Hofmann et al. 2008), it is promising to combine these results with imitation learning so that an effective way to kernelize movement trajectories can be derived.

8 Conclusions

We have proposed a novel formulation of robot movement primitives that incorporates a kernel-based treatment into the process of minimizing the information-loss in imitation learning. Our approach KMP is capable of preserving the probabilistic properties of human demonstrations, adapting trajectories to different unseen situations described by new temporal or spatial requirements and mixing different trajectories. The proposed method was extended to deal with local frames, which provides the robot with reliable extrapolation capabilities. Since KMP is essentially a kernel-based non-parametric approach given a probabilistic reference trajectory, it overcomes several limitations of state-of-the-art methods, being able to learn complex and high dimensional trajectories. Through extensive evaluations in simulations and real robotic systems, we showed that KMP performs well in a wide range of applications such as time-driven movements and human-robot collaboration scenarios.

Acknowledgments

We thank Fares J. Abu-Dakka, Luka Peternel and Martijn J. A. Zeestraten for their help on real robot experiments. We would also like to thank anonymous reviewers for their valuable comments.

References

Amor HB, Neumann G, Kamthe S, Kroemer O and Peters J (2014) Interaction primitives for human-robot cooperation tasks. In: *Proceedings of IEEE International Conference on Robotics and Automation*, 2831-2837.

Bishop CM (2006) *Pattern Recognition and Machine Learning*, Chapter 3 and 7, Springer.

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

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

Calinon S and Lee D (2017) *Learning Control*. Vadakkepat, P. and Goswami, A. (eds.). Humanoid Robotics: a Reference. Springer.

Calinon S, Guenter F and Billard A (2007) On learning, representing, and generalizing a task in a humanoid robot. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* 37(2):286-298.

Cohn DA, Ghahramani Z and Jordan MI (1996) Active learning with statistical models. *Journal of Artifial Intelligence Research* 4(1):129-145.

Dong J, Mukadam M, Dellaert F and Boots B (2016) Motion planning as probabilistic inference using gaussian processes and factor graphs. In: *Proceedings of Robotics: Science and Systems*.

Deisenroth M and Rasmussen CE (2011) PILCO: A model-based and data-efficient approach to policy search. In: *Proceedings of the 28th International Conference on machine learning*, 465-472.

Englert P, Paraschos A, Deisenroth MP and Peters J (2013) Probabilistic model-based imitation learning. *Adaptive Behavior* 21(5):388-403.

Ewerton M, Neumann G, Lioutikov R, Amor HB, Peters J and Maeda G (2015) Learning multiple collaborative tasks with a mixture of interaction primitives. In: *Proceedings of IEEE International Conference on Robotics and Automation*, 1535-1542.

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

Guenter F, Hersch M, Calinon S and Billard A (2007) Reinforcement learning for imitating constrained reaching movements. *Advanced Robotics* 21(13):1521-1544.

Havoutis I and Calinon S (2017) Supervisory teleoperation with online learning and optimal control. In: *Proceedings of IEEE International Conference on Robotics and Automation*, 1534-1540.

Hofmann T, Schölkopf B and Smola AJ (2008) Kernel methods in machine learning. *The Annals of Statistics* 1171-1220.

Huang Y, Büchler D, Koç O, Schölkopf B and Peters J (2016) Jointly learning trajectory generation and hitting point prediction in robot table tennis. In: *Proceedings of IEEE International Conference on Humanoid Robots*, 650-655.

Huang Y, Silvério J, Rozo L and Caldwell DG (2018) Generalized task-parameterized skill learning. In: *Proceedings of IEEE International Conference on Robotics and Automation*, 5667-5674.

Ijspeert AJ, Nakanishi J, Hoffmann H, Pastor P and Schaal S (2013) Dynamical movement primitives: learning attractor models for motor behaviors. *Neural Computation* 25(2):328-373.

Kahn G, Zhang T, Levine S and Abbeel P (2017) Plato: policy learning using adaptive trajectory optimization. In: *Proceedings of IEEE International Conference on Robotics and Automation*, 3342-3349.

Kober J, Öztop E and Peters J (2011) Reinforcement learning to adjust robot movements to new situations. In: *Proceedings of International Joint Conference on Artificial Intelligence*, 2650-2655.

Kullback S and Leibler RA (1951) On information and sufficiency. *The annals of mathematical statistics* 22(1):79-86.

Levine S and Abbeel P (2014) Learning neural network policies with guided policy search under unknown dynamics. In: *Proceedings of Advances in Neural Information Processing Systems*, 1071-1079.

Maeda G, Ewerton M, Neumann G, Lioutikov R and Peters J (2017) Phase estimation for fast action recognition and trajectory generation in humanrobot collaboration. *The International Journal of Robotics Research* 36(13-14):1579-1594.

Matsubara T, Hyon SH and Morimoto J (2010) Learning stylistic dynamic movement primitives from multiple demonstrations. In: *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, 1277-1283.

Medina JR, Lee D and Hirche S (2012) Risk-sensitive optimal feedback control for haptic assistance. In: *Proceedings of IEEE International Conference on Robotics and Automation*, 1025-1031.

Muhlig M, Gienger M, Hellbach S, Steil JJ and Goerick C (2009) Task-level imitation learning using variance-based movement optimization. In: *Proceedings of IEEE International Conference on Robotics and Automation*, 1177-1184.

Murphy KP (2012) *Machine Learning: A Probabilistic Perspective*. Chapter 14.4.3, pp. 492-493, The MIT Press.

Nakanishi J, Morimoto J, Endo G, Cheng G, Schaal S and Kawato M (2004) Learning from demonstration and adaptation of biped locomotion. *Robotics and Autonomous Systems* 47(2):79-91.

Osa T, Esfahani AM, Stolkin R, Lioutikov R, Peters J and Neumann G (2017) Guiding trajectory optimization by demonstrated distributions. *IEEE Robotics and Automation Letters* 2(2), 819-826.

Paraschos A, Daniel C, Peters JR and Neumann G (2013) Probabilistic movement primitives. In: *Proceedings of Advances in Neural Information Processing Systems*, 2616-2624.

Paraschos A, Daniel C, Peters J and Neumann G (2018) Using probabilistic movement primitives in robotics. *Autonomous Robots* 42(3), 529-551.

Paraschos A, Rueckert E, Peters J and Neumann G (2015) Model-free probabilistic movement primitives for physical interaction. In: *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2860-2866.

Pastor P, Hoffmann H, Asfour T and Schaal S (2009) Learning and generalization of motor skills by learning from demonstration. In: *Proceedings of IEEE International Conference on Robotics and Automation*, 763-768.

Pervez A and Lee D (2017) Learning task-parameterized dynamic movement primitives using mixture of GMMs. *Intelligent Service Robotics* 11(1):61-78.

Peters J, Mülling K and Altun Y (2010) Relative entropy policy search. In: *Proceedings of AAAI Conference on Artificial Intelligence*, 1607-1612.

Peters J, Vijayakumar S and Schaal S (2005) Natural actor-critic. In: *Proceedings of European Conference on Machine Learning*, 280-291.

Rana MA, Mukadam M, Ahmadzadeh SR, Chernova S and Boots B (2017) Towards robust skill generalization: unifying learning from demonstration and motion planning. In: *Proceedings of the Conference on Robot Learning*.

Petersen KB and Pedersen MS (2008) *The matrix cookbook*. Technical University of Denmark.

Rasmussen CE and Williams CK (2006) *Gaussian processes for machine learning*. Chapter 2, Appendix A.5, Cambridge: MIT press.

Rozo L, Bruno D, Calinon S and Caldwell DG (2015) Learning optimal controllers in human-robot cooperative transportation tasks with position and force constraints. In: *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, 1024-1030.

Rozo L, Jiménez P and Torras C (2013) A robot learning from demonstration framework to perform force-based manipulation tasks. *Intelligent Service Robotics* 6(1):33-51.

Saunders C, Gammerman A and Vovk V (1998) Ridge regression learning algorithm in dual variables. In: *Proceedings of International Conference on Machine Learning*, 515-521.

Stulp F and Sigaud O (2013) Robot skill learning: from reinforcement learning to evolution strategies. *Paladyn, Journal of Behavioral Robotics* 4(1):49-61.

Appendices

A Gaussian Mixture Regression (GMR)

Let us write the joint probability distribution $\mathcal{P}(\mathbf{s}, \boldsymbol{\xi})$ as $\begin{bmatrix}\mathbf{s}\\ \boldsymbol{\xi}\end{bmatrix} \sim \sum_{c=1}^{C} \pi_c \mathcal{N}(\boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c)$, and decompose the mean $\boldsymbol{\mu}_c$ and covariance $\boldsymbol{\Sigma}_c$ of the c-th Gaussian component as

$$\boldsymbol{\mu}_c = \begin{bmatrix}\boldsymbol{\mu}_c^s\\ \boldsymbol{\mu}_c^\xi\end{bmatrix} \quad \text{and} \quad \boldsymbol{\Sigma}_c = \begin{bmatrix}\boldsymbol{\Sigma}_c^{ss} & \boldsymbol{\Sigma}_c^{s\xi}\\ \boldsymbol{\Sigma}_c^{\xi s} & \boldsymbol{\Sigma}_c^{\xi\xi}\end{bmatrix}, \tag{51}$$

where the superscripts s and ξ correspond to the input and output variables, respectively.

For a query input $\mathbf{s}_n$, the mean of its corresponding output is computed by

$$\hat{\boldsymbol{\mu}}_n = \mathbb{E}(\hat{\boldsymbol{\xi}}_n|\mathbf{s}_n) = \sum_{c=1}^{C} h_c(\mathbf{s}_n)\boldsymbol{\mu}_c(\mathbf{s}_n) \tag{52}$$

with $h_c(\mathbf{s}_n) = \frac{\pi_c \mathcal{N}(\mathbf{s}_n|\boldsymbol{\mu}_c^s, \boldsymbol{\Sigma}_c^{ss})}{\sum_{k=1}^{C} \pi_k \mathcal{N}(\mathbf{s}_n|\boldsymbol{\mu}_k^s, \boldsymbol{\Sigma}_k^{ss})}$ and $\boldsymbol{\mu}_c(\mathbf{s}_n) = \boldsymbol{\mu}_c^\xi + \boldsymbol{\Sigma}_c^{\xi s}(\boldsymbol{\Sigma}_c^{ss})^{-1}(\mathbf{s}_n - \boldsymbol{\mu}_c^s)$. The corresponding conditional covariance is

$$\hat{\boldsymbol{\Sigma}}_n = \mathbb{D}(\hat{\boldsymbol{\xi}}_n|\mathbf{s}_n) = \mathbb{E}(\hat{\boldsymbol{\xi}}_n\hat{\boldsymbol{\xi}}_n^{\mathsf{T}}|\mathbf{s}_n) - \mathbb{E}(\hat{\boldsymbol{\xi}}_n|\mathbf{s}_n)\mathbb{E}^{\mathsf{T}}(\hat{\boldsymbol{\xi}}_n|\mathbf{s}_n). \tag{53}$$

with $\mathbb{E}(\hat{\boldsymbol{\xi}}_n\hat{\boldsymbol{\xi}}_n^{\mathsf{T}}|\mathbf{s}_n) = \sum_{c=1}^{C} h_c(\mathbf{s}_n)\left(\bar{\boldsymbol{\Sigma}}_c + \boldsymbol{\mu}_c(\mathbf{s}_n)\boldsymbol{\mu}_c(\mathbf{s}_n)^{\mathsf{T}}\right)$ and $\bar{\boldsymbol{\Sigma}}_c = \boldsymbol{\Sigma}_c^{\xi\xi} - \boldsymbol{\Sigma}_c^{\xi s}(\boldsymbol{\Sigma}_c^{ss})^{-1}\boldsymbol{\Sigma}_c^{s\xi}$. Thus, the conditional distribution $\hat{\boldsymbol{\xi}}_n|\mathbf{s}_n \sim \mathcal{N}(\hat{\boldsymbol{\mu}}_n, \hat{\boldsymbol{\Sigma}}_n)$ is determined.

B Interpretation of Mean Minimization Problem

On the basis of the definition of multivariate Gaussian distribution, we have

$$\prod_{n=1}^{N}\mathcal{P}(\boldsymbol{\Theta}(\mathbf{s}_n)^{\mathsf{T}}\boldsymbol{\mu}_w|\hat{\boldsymbol{\mu}}_n,\hat{\boldsymbol{\Sigma}}_n)=\prod_{n=1}^{N}\frac{1}{(2\pi)^{\mathcal{O}/2}|\hat{\boldsymbol{\Sigma}}_n|^{1/2}} exp\left\{-\frac{1}{2}(\boldsymbol{\Theta}(\mathbf{s}_n)^{\mathsf{T}}\boldsymbol{\mu}_w-\hat{\boldsymbol{\mu}}_n)^{\mathsf{T}}\hat{\boldsymbol{\Sigma}}_n^{-1}(\boldsymbol{\Theta}(\mathbf{s}_n)^{\mathsf{T}}\boldsymbol{\mu}_w-\hat{\boldsymbol{\mu}}_n)\right\}, \tag{54}$$

which can be further simplified using the logarithmic transformation, yielding

$$\sum_{n=1}^{N}\log\mathcal{P}(\boldsymbol{\Theta}(\mathbf{s}_n)^{\mathsf{T}}\boldsymbol{\mu}_w|\hat{\boldsymbol{\mu}}_n,\hat{\boldsymbol{\Sigma}}_n)=\sum_{n=1}^{N}-\frac{1}{2}(\boldsymbol{\Theta}(\mathbf{s}_n)^{\mathsf{T}}\boldsymbol{\mu}_w-\hat{\boldsymbol{\mu}}_n)^{\mathsf{T}}\hat{\boldsymbol{\Sigma}}_n^{-1}(\boldsymbol{\Theta}(\mathbf{s}_n)^{\mathsf{T}}\boldsymbol{\mu}_w-\hat{\boldsymbol{\mu}}_n)+constant. \tag{55}$$

Thus, the mean minimization problem described by (10) can be interpreted as the maximization of the posterior defined in (54),

C Proof of Weighted Mean Minimization Subproblem

The derivative of (37) with respect to $\boldsymbol{\mu}_w$ can be computed as

$$\begin{aligned}\frac{\partial J_{ini}^{S}(\boldsymbol{\mu}_w)}{\partial\boldsymbol{\mu}_w}&=\sum_{n=1}^{N}\sum_{l=1}^{L}2\left\{\boldsymbol{\Theta}(\mathbf{s}_n)\left(\frac{\hat{\boldsymbol{\Sigma}}_{n,l}}{\gamma_{n,l}}\right)^{-1}\boldsymbol{\Theta}^{\mathsf{T}}(\mathbf{s}_n)\boldsymbol{\mu}_w-\boldsymbol{\Theta}(\mathbf{s}_n)\left(\frac{\hat{\boldsymbol{\Sigma}}_{n,l}}{\gamma_{n,l}}\right)^{-1}\hat{\boldsymbol{\mu}}_{n,l}\right\}\\&=\sum_{n=1}^{N}2\boldsymbol{\Theta}(\mathbf{s}_n)\left\{\sum_{l=1}^{L}\left(\frac{\hat{\boldsymbol{\Sigma}}_{n,l}}{\gamma_{n,l}}\right)^{-1}\right\}\boldsymbol{\Theta}^{\mathsf{T}}(\mathbf{s}_n)\boldsymbol{\mu}_w-\sum_{n=1}^{N}2\boldsymbol{\Theta}(\mathbf{s}_n)\left\{\sum_{l=1}^{L}\left(\frac{\hat{\boldsymbol{\Sigma}}_{n,l}}{\gamma_{n,l}}\right)^{-1}\hat{\boldsymbol{\mu}}_{n,l}\right\}.\end{aligned} \tag{56}$$

Using the definitions (41) and (42), we have

$$\frac{\partial J_{ini}^{S}(\boldsymbol{\mu}_w)}{\partial\boldsymbol{\mu}_w}=\sum_{n=1}^{N}2\boldsymbol{\Theta}(\mathbf{s}_n)\boldsymbol{\Sigma}_n^{S^{-1}}\boldsymbol{\Theta}^{\mathsf{T}}(\mathbf{s}_n)\boldsymbol{\mu}_w-\sum_{n=1}^{N}2\boldsymbol{\Theta}(\mathbf{s}_n)\boldsymbol{\Sigma}_n^{S^{-1}}\boldsymbol{\mu}_n^{S}=\frac{\partial\tilde{J}_{ini}^{S}(\boldsymbol{\mu}_w)}{\partial\boldsymbol{\mu}_w}. \tag{57}$$

Thus, we conclude that the minimization problem in (37) is equivalent to the problem described by (39).

D Proof of Weighted Variance Minimization Subproblem

The derivative of (38) with respect to $\boldsymbol{\Sigma}_w$ is

$$\begin{aligned}\frac{\partial J_{ini}^{S}(\boldsymbol{\Sigma}_w)}{\partial\boldsymbol{\Sigma}_w}&=\sum_{n=1}^{N}\sum_{l=1}^{L}\left\{-\gamma_{n,l}\boldsymbol{\Sigma}_w^{-1}+\gamma_{n,l}\boldsymbol{\Theta}(\mathbf{s}_n)\hat{\boldsymbol{\Sigma}}_{n,l}^{-1}\boldsymbol{\Theta}(\mathbf{s}_n)^{\mathsf{T}}\right\}\\&=\sum_{n=1}^{N}\left\{\sum_{l=1}^{L}-\gamma_{n,l}\boldsymbol{\Sigma}_w^{-1}\right\}+\sum_{n=1}^{N}\boldsymbol{\Theta}(\mathbf{s}_n)\left\{\sum_{l=1}^{L}\left(\frac{\hat{\boldsymbol{\Sigma}}_{n,l}}{\gamma_{n,l}}\right)^{-1}\right\}\boldsymbol{\Theta}(\mathbf{s}_n)^{\mathsf{T}}\\&=\sum_{n=1}^{N}\{-\boldsymbol{\Sigma}_w^{-1}\}+\sum_{n=1}^{N}\boldsymbol{\Theta}(\mathbf{s}_n)\boldsymbol{\Sigma}_n^{S^{-1}}\boldsymbol{\Theta}(\mathbf{s}_n)^{\mathsf{T}}\\&=\frac{\partial\tilde{J}_{ini}^{S}(\boldsymbol{\Sigma}_w)}{\partial\boldsymbol{\Sigma}_w},\end{aligned} \tag{58}$$

so we have proved that the minimization problem in (38) has the same solution as the problem defined in (40).