Task-Adaptive Robot Learning from Demonstration with Gaussian Process Models under Replication

Miguel Arduengo[1], Adrià Colomé[1], Júlia Borràs[1], Luis Sentis[2] and Carme Torras[1]

***Abstract*—Learning from Demonstration (LfD) is a paradigm that allows robots to learn complex manipulation tasks that can not be easily scripted, but can be demonstrated by a human teacher. One of the challenges of LfD is to enable robots to acquire skills that can be adapted to different scenarios. In this paper, we propose to achieve this by exploiting the variations in the demonstrations to retrieve an adaptive and robust policy, using Gaussian Process (GP) models. Adaptability is enhanced by incorporating task parameters into the model, which encode different specifications within the same task. With our formulation, these parameters can be either real, integer, or categorical. Furthermore, we propose a GP design that exploits the structure of replications, i.e., repeated demonstrations with identical conditions within data. Our method significantly reduces the computational cost of model fitting in complex tasks, where replications are essential to obtain a robust model. We illustrate our approach through several experiments on a handwritten letter demonstration dataset.**

***Index Terms*—Learning from Demonstration, Probability and Statistical Methods, Human-Centered Robotics.**

I. INTRODUCTION

LEARNING from Demonstration (LfD) is the paradigm in which robots implicitly learn task constraints from demonstrations. This allows more intuitive skill transfer, satisfying a need of opening policy development to non-robotic-experts as robots extend to assistive domains. The choice of LfD is particularly compelling when ideal behavior can be neither scripted nor easily defined as a reward function but can be demonstrated (Figure 1). One of the fundamental questions is *What to imitate?* [1]. Trajectory-learning methods are usually adopted since they allow a direct skill transfer to robot actions, at both joint, and task space levels. Learning a manipulation task at a trajectory level involves modeling the set of demonstrated motions and retrieving a generalized representation. Among the most relevant contributions in the field over the past decade, we can highlight the approaches based on Dynamic Movement Primitives (DMP) [2], Probabilistic Movement Primitives (ProMP) [3], Gaussian Mixture Models (GMM) [4], Kernelized Movement Primitives (KMP) [5] and Gaussian Process models (GP) [6]. In a recent work [7], we presented a GP-based LfD framework, which we adopt as a basis for this paper. For a comparison with the aforementioned approaches, the reader can refer to our work.

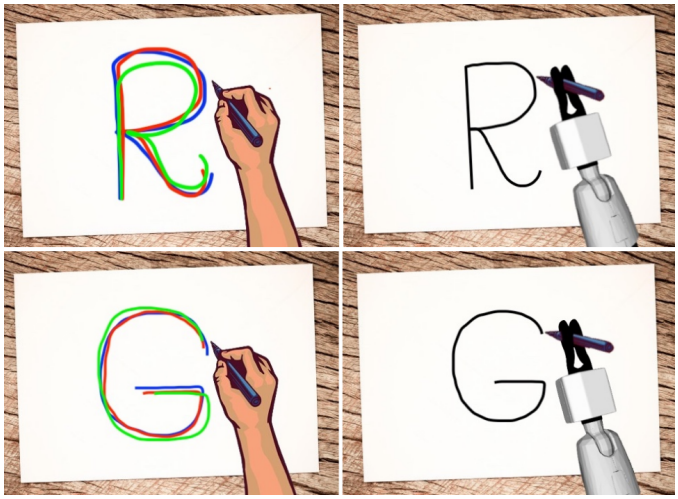


Fig. 1. Learning from Demonstration allows transferring the robot skills that can be intuitively demonstrated but are difficult to script, such as handwriting. On the left, the demonstrations. On the right, the learned policy.

The focus of this paper is on the generalization performance of the learned policy at a task level with GP models. There are skills, in which multiple demonstrations of the same task can look very different due to the task-specific variations. This variability can be interpreted to be governed by the so-called *task variables*, which can describe the current context or a particular requirement. In the LfD literature, generalization has been mainly achieved with two distinct approaches: (a) encoding the demonstrations from the perspective of multiple reference frames; (b) considering task variables as inputs to the learned movement policy, enabling the generation of a path adapted to the context. Approach (a) is motivated by the observation that skillful movement planning often requires the orchestration of multiple coordinate systems that can have varying levels of importance along the task. In [4], the authors propose Task-Parametrized GMM (TP-GMM), a direct extension of GMM. By providing a set of candidate frames that can be potentially relevant, a local statistical analysis is conducted to learn how to retrieve a general trajectory that is invariant to translations and rotations. The idea presented in TP-GMM is also applied to KMP in [8]. Using local coordinate systems, the robot is able to learn about the superposition and transition between the reference frames, resulting in improved extrapo-

Manuscript received: October, 15, 2020; Revised December, 16, 2020; Accepted January, 22, 2021.

This paper was recommended for publication by Editor Dana Kulic upon evaluation of the Associate Editor and Reviewers' comments. This work has been partially funded by the European Union Horizon 2020 Programme under grant agreement no. 741930 (CLOTHILDE) and by the Spanish State Research Agency through the María de Maeztu Seal of Excellence to IRI [MDM-2016-0656]..

[1]First Author, Second Author, Third Author and Fifth Author are with Institut de Robòtica i Informàtica Industrial, CSIC-UPC (IRI), Barcelona. {marduengo, acolome, jborras, torras}@iri.upc.edu

[2] Fourth Author is with the Human Centered Robotics Laboratory, University of Texas at Austin, Austin. lsentis@austin.utexas.edu

lation capabilities. This generalization performance comes at the expense of limiting the task parameters to be in the form of coordinate systems or local projection operators. On the other hand, approach (b) has been applied to DMP, ProMP, and GP. In [9], the authors present stylistic DMP (SDMP), which allows a compact encoding of diverse styles in the demonstrations by including a real-valued control variable in the model, called style parameter. Regarding ProMP, in [3], the authors propose to adapt the primitive based on an external real-valued state variable by learning a linear mapping to the mean weight vector. Finally, in [10], the authors model a map, from a target in 3-D space, to the reaching trajectory, with demonstrations using a GP model. Although (b) provides good interpolation results, the performance far from the regions covered by the demonstrations is limited. However, this approach is generic since the task variables can represent arbitrary context features.

Additionally, one should keep in mind that LfD is a supervised learning approach. Although extrapolation capabilities can be enhanced, providing enough demonstrations to cover the action space is essential. When the model is probabilistic (ProMP, GMM, KMP and GP), repetitions are also required for adequately inferring the statistics of the taught motion. That is, the demonstration dataset must sample the task variable space as densely as possible, providing as many replications as feasible. Thus, for modeling complex manipulation tasks the amount of required data might increase considerably. Commonly, LfD methods are applied with a dataset of a few demonstrations, since they do not scale very well. On the one hand, ProMP and GMM involve an Expectation-Maximization algorithm. On the other hand, KMP and GP require a matrix inversion, whose dimension increases with the number of data points. In this paper, we focus on movement policies learned by demonstration using GP. The approaches for improving the scalability of GP while retaining favorable prediction quality can be divided into two groups: sparse (or global) approximations [11], where the training dataset is approximated by a smaller set of so called support points; and local approximations [12], where the dataset is divided into subsets, using only points near the desired input location for making the prediction. The main drawback of these solutions is their approximate nature.

The main contributions of the paper are two-fold: (1) a GP design for including task variables in the model, which can be either real, integer or categorical; (2) the exploitation of the structure of replications, for alleviating the computational complexity of GP while retrieving an exact model. Our aim is to extend our previous work on LfD with GP by enhancing its generalization to variant task conditions and its capability for scaling with a large demonstration dataset. The structure of the paper is organized as follows: in Section II we formally state the considered LfD problem; in Section III we outline the main theoretical aspects of GP-based LfD; then in Section IV we present our approach for including task variables and exploiting replications; next, in Section V, we illustrate the main concepts and analyze the performance of the proposed solution by learning the writing task from a handwritten letter dataset; finally, in Section VI we summarize the main conclusions.

II. LfD Problem Statement

We formally construct the LfD problem as follows. The robot is presented with a demonstration dataset:

$$\mathcal{D} = \left\{ \boldsymbol{x}_{ij}, \boldsymbol{y}_{ij} \right\}_{i,\, j=1}^{N,\, M_i} \tag{1}$$

where $\boldsymbol{x}_{ij} \in \mathcal{X}$ denotes the task variables and $\boldsymbol{y}_{ij} \in \mathbb{R}^{\mathcal{O}}$ the variables describing the demonstrated motion. Here, the super-indexes N, M_i and $\mathcal{O}$, correspond respectively to the number of demonstrations, training samples per demonstration (which can vary) and the output dimension. Assuming that one of the input components is time t, $\mathcal{X} = \mathbb{R} \times \prod_{i=2}^{\mathcal{I}} \mathbb{X}_i$, being $\mathcal{I}$ the input dimension and $\mathbb{X}_i$ the subspace where the i-th component lies. The LfD problem is solved by learning a task movement policy $\boldsymbol{\pi} : \mathcal{X} \longrightarrow \mathbb{R}^{\mathcal{O}}$ from $\mathcal{D}$, which is capable of inferring the required path to successfully perform the desired manipulation given a new set of task variables. The policy must generalize over multiple demonstrations, and also, in order to be reproduced by the robot, the generated paths have to be continuous and smooth.

III. Gaussian-Process-based LfD

Gaussian Process are a probabilistic representation that allow to encode the underlying trajectory distribution from multiple demonstrations of a manipulation task. For simplicity, in this section we propose to use $\mathcal{D} = \left\{ t_{ij}, \boldsymbol{y}_{ij} \right\}_{i,\, j=1}^{N,\, M_i}$ i.e. the only input component is time. A high-level description of the GP-based LfD framework is summarized in Figure 2. We first perform a preprocessing step for temporally aligning and scaling the demonstrated trajectories. For the training phase, we provide a series of GP design guidelines for modeling robot manipulation tasks from $\mathcal{D}$. Finally, we briefly discuss how the learned policy can be modulated through via-points during the movement execution phase.

A. Preprocessing: Temporal Alignment and Scaling

In general, it is difficult to provide all the demonstrations with the same speed. Time shifts might lead to poor performance when inferring the variability. For aligning temporally the demonstrated trajectories Dynamic Time Warping (DTW) [13] can be used. It is a method that calculates an optimal match, usually nonlinear, with respect to a reference trajectory based on a similarity measure. We use the Task Completion Index (TCI) [7], that is a measure of the portion of the trajectory covered for task completion. For demonstration i, it can be computed as

$$\zeta_{i,k} = \frac{\sum_{j=1}^{k} \| \boldsymbol{y}_{i,j} - \boldsymbol{y}_{i,j-1} \|}{\sum_{j=1}^{M_i} \| \boldsymbol{y}_{i,j} - \boldsymbol{y}_{i,j-1} \|} \qquad \forall k = 1, \ldots, M_i \tag{2}$$

Adjusting the execution speed of the robot is sometimes desirable. As for DMP and ProMP, we can consider a normalized phase variable to decouple the path from the time signal. This is equivalent to scale the time component. Let t_R be the time length of the reference trajectory. To adapt the trajectory to a desired duration t_D, we can define a monotonic increasing function $s : [0, t_R] \longrightarrow [0, t_D]$. Note that this step can also be carried out during the execution.

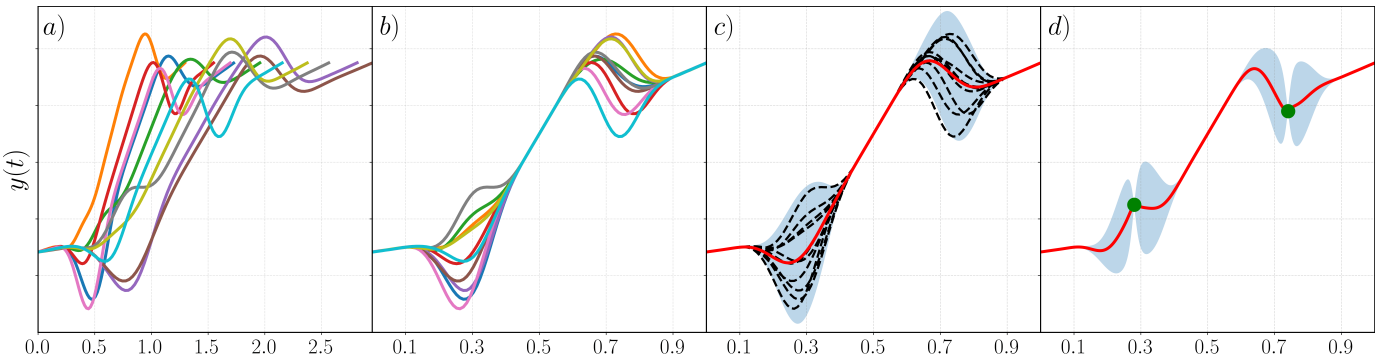


Fig. 2. Learning from demonstration with Gaussian Processes. **(a)** Several demonstrations of the manipulation task. The trajectories present spatial variability as well as different time lengths. **(b)** They are temporally aligned with the DTW algorithm using the TCI as similarity measure. Also, the time component is scaled, so the trajectory is adapted to the desired time law. **(c)** The movement policy is encoded with a Heteroscedastic GP model. This probabilistic representation effectively encodes the variability (blue shaded area) of the demonstrations (black dashed line) and retrieves a generalized form of the task motion (red solid line). **(d)** The learned policy can be modulated through via-points (green dots) by conditioning the policy on the new points.

B. Robot Task Representation with GP

The task policy must encode the variability as well as generalize over multiple demonstrations of the learned manipulation task. This requires an adequate GP design.

1) Gaussian Process Models: Intuitively, a GP defines a prior over functions, which can be converted into a posterior given a set of input-output pairs [14]. Considering scalar outputs y, it is defined by the scalar mean $m(t)$ and covariance $k(t,t')$ (i.e. kernel) functions, which encode the assumptions on the policy being learned

$$y(t) \sim \mathcal{GP}\left(m(t), k\left(t,t'\right)\right) \tag{3}$$

2) Heteroscedastic GP: The task policy i.e. the path $\boldsymbol{y}^*$ that performs the taught manipulation given a new set of task variables $\boldsymbol{t}^*$ and $\mathcal{D}$, is modeled with GP as a multivariate Gaussian distribution $p(\boldsymbol{y}^*|\boldsymbol{t}^*,\mathcal{D}) \sim \mathcal{N}(\boldsymbol{\mu}^*,\boldsymbol{\Sigma}^*)$. The constant noise level considered in the standard GP formulation can be an important limitation for capturing the variability in the demonstrations, as there will be parts in the task where it might vary. The uniform noise assumption can be relaxed by considering a normally distributed noise $\varepsilon \sim \mathcal{N}(0, r(t))$, where the variance is input-dependent and modelled by the latent function $r()$. The joint distribution of the training $\boldsymbol{y}$, and predicted $\boldsymbol{y}^*$ outputs according to the prior is then

$$\begin{bmatrix} \boldsymbol{y} \\ \boldsymbol{y}^* \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \boldsymbol{m} \\ \boldsymbol{m}^* \end{bmatrix}, \begin{bmatrix} \boldsymbol{K}+\boldsymbol{R} & \boldsymbol{K}^* \\ \boldsymbol{K}^{*T} & \boldsymbol{K}^{**}+\boldsymbol{R}^* \end{bmatrix}\right) \tag{4}$$

being $\boldsymbol{m}$ and $\boldsymbol{m}^*$ arrays whose elements are function $m()$ evaluated at $\boldsymbol{t}$ and $\boldsymbol{t}^*$ respectively; analogously for diagonal matrices $\boldsymbol{R}$ and $\boldsymbol{R}^*$ with function $r()$; $\boldsymbol{K}^*$ is the Gram matrix of the kernel function $k(,)$ evaluated at all pairs (t,t^*); similarly, $\boldsymbol{K}$ and $\boldsymbol{K}^{**}$. The predictive distribution is then obtained by marginalizing on the demonstrations, resulting the following predictive mean $\boldsymbol{\mu}^*$ and variance $\boldsymbol{\Sigma}^*$

$$\boldsymbol{\mu}^* = \boldsymbol{m}^* + \boldsymbol{K}^{*T}\left(\boldsymbol{K}+\boldsymbol{R}\right)^{-1}\left(\boldsymbol{y}-\boldsymbol{m}\right) \tag{5}$$

$$\boldsymbol{\Sigma}^* = \boldsymbol{K}^{**} + \boldsymbol{R}^* - \boldsymbol{K}^{*T}\left(\boldsymbol{K}+\boldsymbol{R}\right)^{-1}\boldsymbol{K}^* \tag{6}$$

The latent noise function $r()$ is usually not known a-priori. As proposed in [15], first an standard GP can be fit to the demonstrations. Its predictions can be then used to estimate the input-dependent noise empirically. Then, a second GP can be used to model $z(t) = \log[r(t)]$. Let $\mathcal{Z} = \{\boldsymbol{z}, \boldsymbol{z}^*\}$ be the set of noise data. The posterior predictive distribution can be then approximated by

$$p(\boldsymbol{y}^*|\boldsymbol{t}^*,\mathcal{D}) \simeq p(\boldsymbol{y}^*|\boldsymbol{t}^*,\mathcal{D},\mathcal{Z}) \tag{7}$$

where $\mathcal{Z} = \arg\max_{\boldsymbol{z},\boldsymbol{z}^*} p(\boldsymbol{z},\boldsymbol{z}^*|\boldsymbol{t}^*,\mathcal{D})$ is the most likely noise level, which can be determined with Monte Carlo or Expectation-Maximization algorithms.

3) Multi-Output GP (MOGP): The previous concepts can be extended to multiple-output GP (MOGP) by taking a matrix covariance function $\boldsymbol{k}(t,t')$. This can be expressed around the Linear Model of Coregionalization (LMC) [16]

$$\boldsymbol{B} \otimes \boldsymbol{k}\left(t,t'\right) = \begin{bmatrix} B_{11}k_{11}\left(t,t'\right) & \dots & B_{1d}k_{1d}\left(t,t'\right) \\ \vdots & \ddots & \vdots \\ B_{d1}k_{d1}\left(t,t'\right) & \dots & B_{dd}k_{dd}\left(t,t'\right) \end{bmatrix} \tag{8}$$

where d is the output dimension and $\boldsymbol{B}$ is the coregionalization matrix. The off-diagonal elements encode output relatedness. In the general case, for learning robot task motions, we cannot make any a-priori assumptions in this regard. Therefore, we can set $B_{ij} = 0$ for $i \neq j$, being the MOGP equivalent to d independent GP.

4) Kernel: In order to be a valid kernel function, the corresponding Gram matrix $\boldsymbol{K}$, with elements $K_{ij} = k(t_i, t_j)$ must be positive semidefinite. Furthermore, the chosen kernel must generate continuous and smooth paths for the robot to be able to execute the motion. Note also that the time parametrization of trajectories is invariant to translations in the time domain. Thus, it should be a function of $\tau = \|t - t'\|$. A popular choice is the squared exponential (SE) kernel

$$k(\tau) = \sigma_f^2 \exp\left(-\frac{\tau^2}{2l^2}\right) \tag{9}$$

where σ_f and l are the hyperparameters. The GP with this covariance function has mean square derivatives of all orders, and is thus very smooth. For slightly relaxing the smoothness prior assumption, the Matérn kernel can also be used

$$k(\tau) = \sigma_f^2\left(1 + \frac{\sqrt{5}\tau}{l} + \frac{5\tau^2}{3l^2}\right)\exp\left(-\frac{\sqrt{5}\tau}{l}\right) \tag{10}$$

For selecting the kernel hyperparameters $\boldsymbol{\Theta}$, the following log marginal likelihood is usually maximized

$$\log p\left(\boldsymbol{y}|\boldsymbol{t},\boldsymbol{\Theta}\right) = -\frac{1}{2}\boldsymbol{y}^T\left(\boldsymbol{K}+\boldsymbol{R}\right)^{-1}\boldsymbol{y} - \frac{1}{2}\log|\boldsymbol{K}+\boldsymbol{R}| \tag{11}$$

This optimization problem might suffer from multiple local optima. Gradient-based or stochastic optimization methods can be used for computing the solution.

C. Policy Modulation with GP

We can modulate the task policy by adapting the learned path to pass trough new via-points $\mathcal{V} = \{t_i, \boldsymbol{y}_i^v\}_{i=1}^{M_v}$. Modulation can be achieved by conditioning the policy on both, $\mathcal{D}$ and $\mathcal{V}$. Assuming that the predictive distribution of each set can be computed independently

$$p\left(\boldsymbol{y}^*|\boldsymbol{t}^*,\mathcal{D},\mathcal{V}\right) \propto p\left(\boldsymbol{y}^*|\boldsymbol{t}^*,\mathcal{D}\right)p\left(\boldsymbol{y}^*|\boldsymbol{t}^*,\mathcal{V}\right) \tag{12}$$

Then, if $p\left(\boldsymbol{y}^*|\boldsymbol{t}^*,\mathcal{D}\right) \sim \mathcal{N}\left(\boldsymbol{\mu}^d,\boldsymbol{\Sigma}^d\right)$ and $p\left(\boldsymbol{y}^*|\boldsymbol{t}^*,\mathcal{V}\right) \sim \mathcal{N}\left(\boldsymbol{\mu}^v,\boldsymbol{\Sigma}^v\right)$, it holds $p\left(\boldsymbol{y}^*|\boldsymbol{t}^*,\mathcal{D},\mathcal{V}\right) \sim \mathcal{N}\left(\boldsymbol{\mu}^{**},\boldsymbol{\Sigma}^{**}\right)$ with

$$\boldsymbol{\mu}^{**} = \boldsymbol{\Sigma}^v\left(\boldsymbol{\Sigma}^d+\boldsymbol{\Sigma}^v\right)^{-1}\boldsymbol{\mu}^d + \boldsymbol{\Sigma}^d\left(\boldsymbol{\Sigma}^d+\boldsymbol{\Sigma}^v\right)^{-1}\boldsymbol{\mu}^v \tag{13}$$

$$\boldsymbol{\Sigma}^{**} = \boldsymbol{\Sigma}^d\left(\boldsymbol{\Sigma}^d+\boldsymbol{\Sigma}^v\right)^{-1}\boldsymbol{\Sigma}^v \tag{14}$$

The resulting policy ponders the demonstrations and the via-points weighted inversely by their variances. Modelling $p\left(\boldsymbol{y}^*|\boldsymbol{t}^*,\mathcal{V}\right)$ with an heteroscedastic GP, the strength of the via-point constraints can be easily specified by means of $r()$. Note that modulation can be performed during the execution of the motion, since $p\left(\boldsymbol{y}^*|\boldsymbol{t}^*,\mathcal{D}\right)$ can be pre-computed.

IV. Task-Adaptive GP under Replication

Our novel GP-based framework takes advantage of the versatility and expressiveness of GP to encompass the main features required for a state-of-the-art LfD approach [7]. Generalization of the learned policy can be enhanced by incorporating task variables that describe the context under which demonstrations are performed. We propose a new GP design inspired by [17], which allows these variables to be either real, integer or categorical, and exploits the possible correlations. Also, for complex tasks, the number of required demonstrations for sampling the action space might increase considerably. This poses a challenge for GP, which suffer from cubic complexity to data size. Motivated by [18], we propose a formulation that exploits the structure of replications, which arise naturally in the LfD context, for achieving significant computational savings.

A. Generalization with Task Variables

Encoding the policy with GP, we can consider task variables as inputs. In this way, relying on $\mathcal{D}$, the model can learn the constraints and requirements of the manipulation task from a wider perspective, being capable of retrieving an adaptive motion for a previously unseen context, described by a new set of task variables. However, the standard GP problem considers only continuous input variables, limiting the applicability of the method in tasks with discrete integer or categorical variables (e.g. object class, house room).

As a first approach, we could fit distinct GP models for each possible combination of the discrete variables. However, this method ignores possible correlations and becomes infeasible as the quantity of discrete sets grows, since the number of models increases exponentially. Another possibility is one-hot encoding i.e. adding as many extra input variables as different values the discrete variable can take. Although it might be appealing for its simplicity, the input dimension can increase dramatically.

Without loss of generality, we consider a three-dimensional input variable $\boldsymbol{x} = (t, s, u)$, being t continuous, s integer and u categorical variables. Thus, we study GP models defined on a finite subspace of $\mathcal{X} = \mathbb{R}\times\mathbb{Z}\times\mathbb{K}$. By focusing the modeling effort on the covariance structure, kernels on $\mathcal{X}$ can be obtained by combining kernels on $\mathbb{R}$, $\mathbb{Z}$ and $\mathbb{K}$. Standard valid combinations are the (1) product, (2) sum or (3) ANOVA. If $k_{\mathbb{X}}$ denotes a kernel for variables that lie in domain $\mathbb{X}$, examples of valid kernels are

1) $k\left(\boldsymbol{x},\boldsymbol{x}'\right) = k_{\mathbb{R}}\left(t,t'\right)k_{\mathbb{Z}}\left(s,s'\right)k_{\mathbb{K}}\left(u,u'\right)$
2) $k\left(\boldsymbol{x},\boldsymbol{x}'\right) = k_{\mathbb{R}}\left(t,t'\right)+k_{\mathbb{Z}}\left(s,s'\right)+k_{\mathbb{K}}\left(u,u'\right)$
3) $k\left(\boldsymbol{x},\boldsymbol{x}'\right) = \left[1+k_{\mathbb{R}}\left(t,t'\right)\right]\left[1+k_{\mathbb{Z}}\left(s,s'\right)\right]\left[1+k_{\mathbb{K}}\left(u,u'\right)\right]$

For $k_{\mathbb{R}}$, kernels such as the the squared exponential (Eq. 9) or Matérn (Eq. 10) can be used. The question then comes down to constructing a valid kernel on a finite subset of $\mathbb{Z}$ or $\mathbb{K}$, with a corresponding positive semidefinite Gram matrix.

1) Kernels for integer variables: An integer variable is a discrete variable with ordered levels. Thus, $\mathbb{Z}$ can be seen as a discretization of $\mathbb{R}$. We can define a non-decreasing transformation $T:\mathbb{Z}\longrightarrow\mathbb{R}$ (i.e. warping) such that the order is preserved, which projects the discrete variable into a continuous space. Consequently, the kernel function can be written as

$$k_{\mathbb{Z}}\left(s,s'\right) = k_{\mathbb{R}}\left(T\left(s\right),T\left(s'\right)\right) \tag{15}$$

In the general case, T is piecewise-linear. However, common warping functions are based on the cumulative distribution of a uniform, normal or lognormal random variable $T:\mathbb{Z}\longrightarrow[0,1]$. Note that when $k_{\mathbb{R}}$ depends on the distance $\|t-t'\|$, then $k_{\mathbb{Z}}$ depends on the distance between s, s' distorted by T. Selecting an appropriate T may require subject-matter knowledge.

In order to allow negative correlations, alternatively to standard SE or Matérn kernels, one may choose, for instance, the cosine correlation kernel on $[0,\beta)$, being $\beta\in(0,\pi]$ a fixed parameter tuning the minimal correlation value

$$k_{\mathbb{Z}}\left(s,s'\right) = \cos\left(T\left(s\right)-T\left(s'\right)\right) \tag{16}$$

2) Kernels for categorical variables: For categorical variables there is no notion of order. However, what does exist is a notion of equality ($=$) or inequality ($\neq$). Among the parsimonious kernel parametrizations, up to additional assumptions, which generate a positive-definite Gram matrix is the compound symmetry (CS) covariance structure

$$k_{\mathbb{K}}\left(u,u'\right) = \begin{cases} v & \text{if } u = u' \\ c & \text{if } u \neq u' \end{cases} \tag{17}$$

where v is the variance and c the covariance. This structure is a generalization of the SE kernel using the Gower distance. All pairs of categories are treated equally, being the similarity maximum for two equal input points, and minimum for different ones. A more flexible parametrization can be obtained by considering groups of categories. Let the discrete categorical set be partitioned into G groups, and $g(u)$ the group number corresponding to value u

$$k_{\mathbb{K}}(u,u') = \begin{cases} v & \text{if } u = u' \\ c_{g(u),g(u')} & \text{if } u \neq u' \end{cases} \tag{18}$$

where for all $i,j \in \{1,\dots,G\}$, the terms $c_{i,i}/v$ are within-group correlations, and $c_{i,j}/v$ $(i \neq j)$ are between-group correlations. Note that additional constraints on v and $c_{i,j}$ are required to ensure that $k_{\mathbb{K}}$ is a valid kernel function. The corresponding Gram matrix $\boldsymbol{K}$, written in block form is

$$\boldsymbol{K}(\boldsymbol{u},\boldsymbol{u}') = \begin{pmatrix} \boldsymbol{W}_1 & \dots & \boldsymbol{B}_{1,G} \\ \vdots & \ddots & \vdots \\ \boldsymbol{B}_{G,1} & \dots & \boldsymbol{W}_G \end{pmatrix} \tag{19}$$

where diagonal blocks $\boldsymbol{W}_g$ and off-diagonal blocks $\boldsymbol{B}_{g,g'}$ encode within-group and between-group covariances respectively. The necessary and sufficient conditions for $\boldsymbol{K}$ to be positive semidefinite are:

- $\boldsymbol{W}_g$ is positive semidefinite $\forall\, g = 1,\dots,G$
- $\boldsymbol{W}_g - \overline{W}_g \boldsymbol{J}_g$ is positive semidefinite $\forall\, g = 1,\dots,G$

where $\boldsymbol{J}_g$ is a matrix of ones with the size of $\boldsymbol{W}_g$ and $\overline{W}_g$ the average of its elements. Thus, for the kernel function defined in Equation (17), considering a discrete set with L different categories, we can derive the following condition

$$-(L-1)^{-1} v < c < v \tag{20}$$

3) Example: Consider a manipulation task modeled by the following deterministic function of $\boldsymbol{x} = (t,s,u)$

$$y(\boldsymbol{x}) = \begin{cases} -\frac{1}{20} t \cdot s & \text{if } u = \text{lin} \\ \frac{3}{10} \sin\left(\pi\left[5t - \frac{1}{4}\right] - s/5\right) & \text{if } u = \text{sin} \\ \frac{1}{4} \sin\left(\pi\left[3t - \frac{1}{2}\right]\right) e^{-0.8t \cdot s} + \frac{1}{10} & \text{if } u = \text{dsin} \end{cases} \tag{21}$$

with $t \in [0,1]$, $s \in \{1,\dots,5\}$ and $u \in \{\text{lin},\text{sin},\text{dsin}\}$. As depicted in Figure 3, the required path depends greatly on u, while small variations are explained by s.

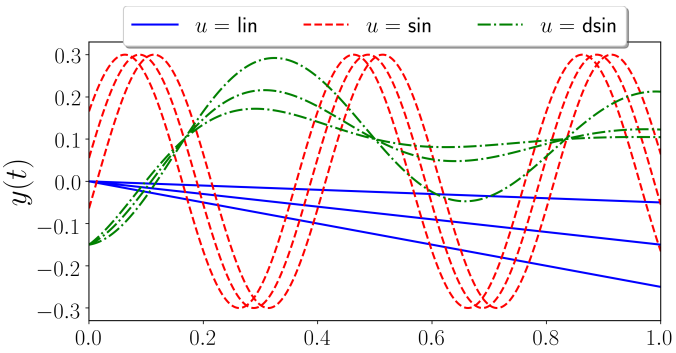
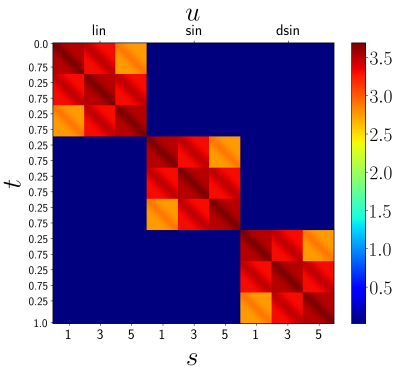


Fig. 3. Projection on the y-t plane of the task model $y(\boldsymbol{x})$ (Eq. 21)



Kernel	R^2
product	0.52
sum	0.37
ANOVA	0.97

Fig. 4. Matrix $\boldsymbol{K}$ constructed with ANOVA composition and maximum likelihood over a train set formed by a regular $8 \times 3 \times 3$ on (t,s,u) domain. Also, R^2 coefficient of the inferred $y(\boldsymbol{x})$ using different compositions.

We aim at reconstructing the task policy from demonstrations. As training data, we take samples on a regular $8 \times 3 \times 3$ grid in the task variable domain $[0,1] \times \{1,\dots,5\}, \times \{\text{lin},\text{sin},\text{dsin}\}$. We consider three models using sum, product and ANOVA kernel combinations. For $k_{\mathbb{R}}$, $k_{\mathbb{Z}}$ and $k_{\mathbb{K}}$ we use the SE (Eq. 9), cosine (Eq. 16) and CS (Eq. 17) kernels respectively. Hyperparameters are estimated by maximum likelihood. Model accuracy is measured in terms of the R^2 criterion over a test set formed by a finer $100 \times 5 \times 3$ grid in order to illustrate the generalization capabilities. In Figure 4 we show the resulting matrix $\boldsymbol{K}$ with ANOVA composition, for which we achieve the best score. We can observe three distinct groups almost uncorrelated for different values of u, each one divided into three subgroups, one for each training s, with high relatedness, as it can be intuited from Figure 3. Thus, we can conclude that the proposed GP design retrives an effective policy by including continous, integer and categorical variables; inferring an accurate covariance structure.

B. Inference and Prediction under Replication

In the LfD context, replications, which can be intuitively defined as repeated demonstrations for identical task variables, play a key role on the estimation of the variability. This constitutes a challenge, since the computational complexity of the GP-based policy learning algorithm increases cubically with the number of training samples. We exploit the structure of replications in GP design for achieving computational savings during inference and prediction by using two well-known formulas, together comprising the Woodbury identity

$$(\boldsymbol{D} + \boldsymbol{UBV})^{-1} = \boldsymbol{D}^{-1} - \boldsymbol{D}^{-1}\boldsymbol{U}\left(\boldsymbol{B}^{-1} + \boldsymbol{V}\boldsymbol{D}^{-1}\boldsymbol{U}\right)^{-1}\boldsymbol{V} \tag{22}$$

$$|\boldsymbol{D} + \boldsymbol{UBV}| = \left|\boldsymbol{B}^{-1} + \boldsymbol{V}\boldsymbol{D}^{-1}\boldsymbol{U}\right| \cdot |\boldsymbol{B}| \cdot |\boldsymbol{D}| \tag{23}$$

Let N and $n \ll N$ be the number of training samples and unique input locations (points with identical task variables) respectively; $y_i^{(j)}$ be the j^{th} out of $a_i \geq 1$ replicates observed at each unique input, where $\sum_{i=1}^{n} a_i = N$; and $\bar{\boldsymbol{y}}$ be the array with concatenated terms $\bar{y}_i = a_i^{-1} \sum_{j=1}^{a_i} y_i^{(j)}$. We now develop a map from full $\boldsymbol{K}_N$, $\boldsymbol{R}_N$ matrices (Eq. 4) to their unique-n counterparts $\boldsymbol{K}_n$, $\boldsymbol{R}_n$. Without loss of generality, assume that data is ordered such that replicates are stacked together. Then

$$\boldsymbol{K}_N = \boldsymbol{U}\boldsymbol{K}_n\boldsymbol{U}^T \qquad \boldsymbol{U}^T\boldsymbol{R}_N\boldsymbol{U} = \boldsymbol{A}_n\boldsymbol{R}_n \tag{24}$$

with $\boldsymbol{U} = \text{diag}\left(\mathbf{1}_{a_1,1}\ldots,\mathbf{1}_{a_n,1}\right)$ a $N \times n$ block matrix, where $\mathbf{1}_{k,l}$ is a $k \times l$ matrix of ones, and $\boldsymbol{A}_n = \text{diag}\left(a_1,\ldots,a_n\right)$. Developing equations 5, 6 and 11, taking $\boldsymbol{D} \equiv \boldsymbol{R}_N$, $\boldsymbol{B} \equiv \boldsymbol{K}_n$ and $\boldsymbol{V} = \boldsymbol{U}^T$, yields the following predictive equations

$$\boldsymbol{\mu}^* = \boldsymbol{m}^* + \boldsymbol{K}_n^{*T}\left(\boldsymbol{K}_n + \boldsymbol{A}_n^{-1}\boldsymbol{R}_n\right)^{-1}\left(\overline{\boldsymbol{y}} - \overline{\boldsymbol{m}}\right) \tag{25}$$

$$\boldsymbol{\Sigma}^* = \boldsymbol{K}^{**} + \boldsymbol{R}^* - \boldsymbol{K}_n^{*T}\left(\boldsymbol{K}_n + \boldsymbol{A}_n^{-1}\boldsymbol{R}_n\right)^{-1}\boldsymbol{K}_n^* \tag{26}$$

Note they are almost identical to the original ones built by overlooking the structure of replication. We also have the next expression for the log likelihood $\log\mathcal{L}$

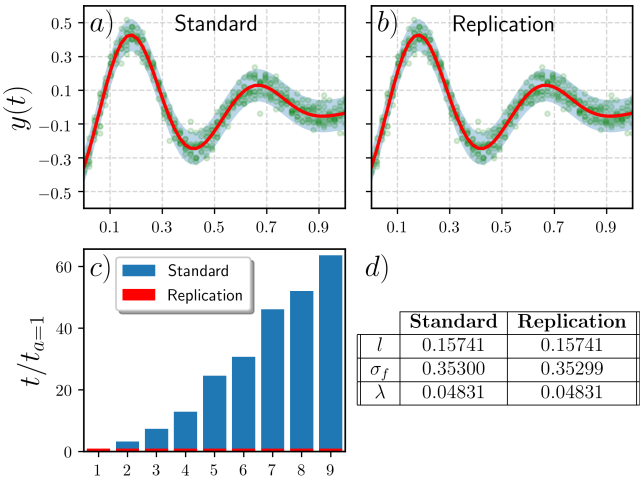
$$\log\mathcal{L} = -\frac{1}{2}\left(\boldsymbol{y}^T\boldsymbol{R}_N^{-1}\boldsymbol{y} - \overline{\boldsymbol{y}}^T\boldsymbol{A}_n\boldsymbol{R}_n^{-1}\overline{\boldsymbol{y}} + \overline{\boldsymbol{y}}^T\left(\boldsymbol{K}_n + \boldsymbol{A}_n^{-1}\boldsymbol{R}_n\right)^{-1}\overline{\boldsymbol{y}}\right) - \frac{1}{2}\left(\log\left|\boldsymbol{K}_n + \boldsymbol{A}_n^{-1}\boldsymbol{R}_n\right| + \log\left|\boldsymbol{R}_N\right| - \log\left|\boldsymbol{A}_n^{-1}\boldsymbol{R}_n\right|\right) \tag{27}$$

This implies that only $\mathcal{O}\left(n^3\right)$ matrix decompositions are required, which could represent huge savings compared to $\mathcal{O}\left(N^3\right)$ if the degree of replication is high.

For illustrating the potential of exploiting the structure of replications in the design, consider a manipulation task with the following observational model

$$y(t) = \sin\left(\pi\left[4t - 0.25\right]\right)e^{-3t}\left(1 + e^{-5x}\right)^{-1} + \varepsilon \tag{28}$$

where $\varepsilon \sim \mathcal{N}\left(0, \lambda = 0.05\right)$. For the model, we assume an SE kernel with hyperparameters l and σ_f, and constant noise level with variance λ, which we also consider a hyperparameter. Samples are taken at $n = 50$ unique input locations, each having a replicates. In Figure 5c, we can observe that for the standard formulation, the computational time for retrieving the policy with respect to $a = 1$ increases dramatically with the number of replicates, whereas it remains constant with the proposed one. For the case $a = 9$, where the time differs by a factor of 60, we can observe in figures 5a and 5b that the learned policy is identical. Also, the inferred hyperparameters (Figure 5d) since the method is exact.



	Standard	Replication
l	0.15741	0.15741
σ_f	0.35300	0.35299
λ	0.04831	0.04831

Fig. 5. (a) Learned policy with the standard GP formulation. The red solid line, blue shaded area and green dots refer respectively to the mean, the 95% confidence interval and the training samples. (b) Same with the replication-based design. (c) Computational time for retrieving the policy with respect to $a = 1$ for different a. (d) Inferred hyperparameters of the SE kernel, l and σ_f, and noise variance λ.

V. Illustrative Example: Learning to Write

In this section, we illustrate and evaluate the main aspects of the presented GP-based LfD framework through the robot writing task. This skill suits perfectly the LfD context since it is difficult to script but can be intuitively demonstrated. The task has been explored for engaging robots in teaching activities. Building up on the learning by teaching paradigm, letting a child demonstrate the robot, not only does the child practice handwriting but, also positively reinforce their motivation [19]. First, we present the robot an experimental handwritten dataset, which includes trajectories for several letters indexed by real, integer and categorical task variables. Then, we consider the problem of learning a movement policy from the demonstrations, comparing the performance of different GP designs. Next, we assess the adaptability of the resulting policy by evaluating the modulation through via-points, and the interpolation and extrapolation capabilities. Finally, we study, in terms of computational time, the implications of considering the structure of replications in the formulation.

A. Handwritten Letter Dataset

The demonstration dataset has been generated experimentally by handwriting different letters on a tablet using a standard note app (Figure 6). We extracted the data by first screen recording while writing, and then, processing the resulting videos with computer vision techniques. As output variables we take the x and y coordinates of the path that describes the handwriting motion. As input (or task) variables we take time t; the size of the letter s, which we consider defined by the height, measured as an integer height $= s \times 8$mm; and finally, the letter corresponding to the motion u, e.g. 'A', 'B' ,'C', etc. The variables and the domain sampled in the dataset are summarized in Table I.

Variable	Symbol	Type	Domain
Time	t	Input	$[0, 1]$
Size	s	Input	$\{2,3,4,5,6\}$
Letter	u	Input	$\{\text{A},\text{B},\text{C},\text{D}\}$
Horizontal coordinate	x	Output	$\mathbb{R}$
Vertical coordinate	y	Output	$\mathbb{R}$

Table I. Handwritten letter dataset variables.

The size of the dataset was guided by the discrete task variables, such that each of the $5 \times 4 = 20$ possible discrete input locations has 5 replicates. That is a total of $N = 100$ different demonstrations. The dataset, after temporal alignment and scaling of trajectories, is shown in Figure 7.

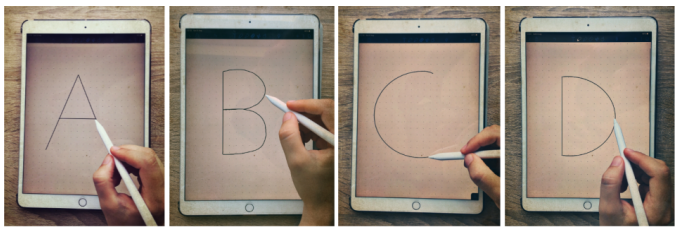


Fig. 6. Demonstrations for the writing task are performed using a tablet.

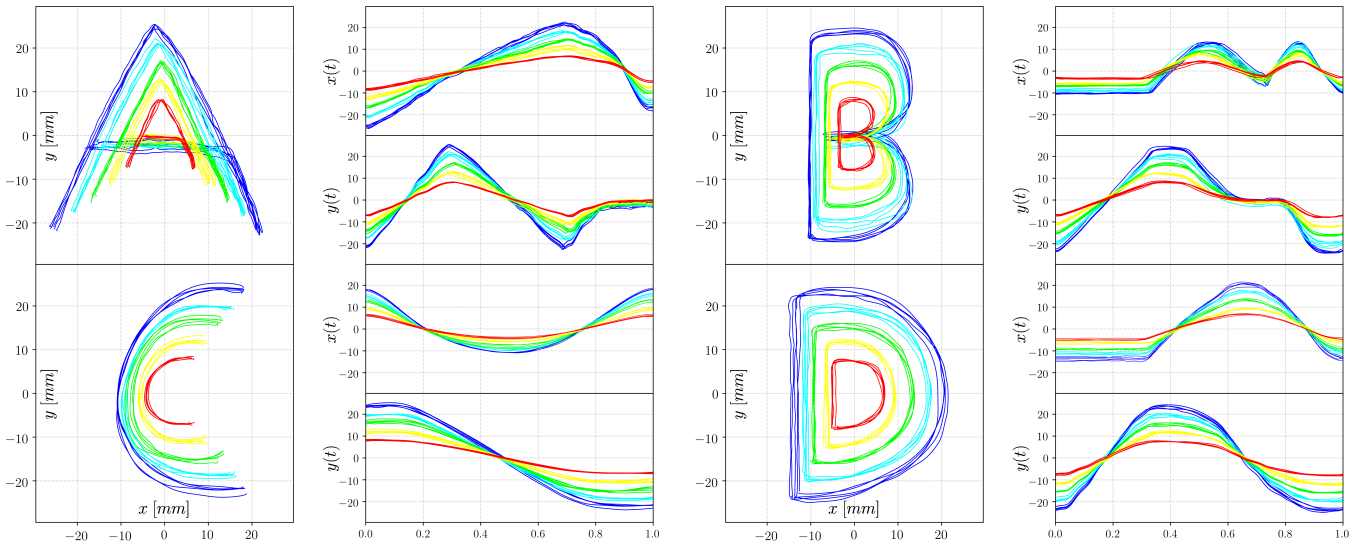


Fig. 7. Handwritten letter dataset. Trajectories are indexed by time t, size s and letter u. 5 replicates are provided for each possible combination of the discrete variables s and u, represented in the same color. To the right of each letter x-y path, the corresponding $x(t)$ and $y(t)$ are shown.

B. Task Model Design

We aim to learn a movement policy from the demonstrations that is capable of generating the motion required to write a given letter with a specified size. In the presented GP approach the modeling effort is focused on the covariance structure. The multi-dimensional kernel function is built as a composition, which can be either sum, product or ANOVA, of one-dimensional ones. For $k_{\mathbb{R}}$ we can use the SE or the Matérn; for $k_{\mathbb{Z}}$ we propose the cosine kernel and, due to the nature of the size variable, a linear transformation $T()$ (Eq. 15); finally, for $k_{\mathbb{K}}$ we take the CS structure. To select the best model, we split the dataset in 50% for learning the task policy and the remaining 50% for assessing its performance. The results obtained for different covariance structures in terms of the R^2 coefficient are shown in Table II. Although very accurate predictions are obtained either with the product or ANOVA composition, the best score is achieved by the ANOVA+Matérn model. In Figure 8a we can see that this policy effectively retrieves the motion required for writing a letter 'A' of size 4, also encoding the variability.

Composition / $k_{\mathbb{R}}$	Product	Sum	ANOVA
SE	0.87	0.32	0.93
Matérn	0.89	0.35	0.94

Table II. R^2 coefficient for different covariance structures.

C. Adaptability of the Task Policy

Here we illustrate the generalization capabilities of the learned task policy. Adaptability can be achieved either through the specification of via-points, or relying on the GP model to generate the required motion given a new set of task parameters, for which demonstrations are not provided. We study different possibilities through the example shown in Figure 8. As depicted in Figure 8b, the motion in Figure 8a can be easily modulated to fulfill new specifications by conditioning the learned distribution to pass through new initial, final and/or intermediate via-points. Now consider that during the execution of the task, the robot encounters a new context, not sampled in the demonstration set. We consider two different cases, in Figure 8c we only use data of letters with sizes 3 and 5, whereas in Figure 8d we only consider sizes 5 and 6. Thus, for writing a letter 'A' of size 4 interpolation is required in the former case, and extrapolation in the latter. Since the new task variables are close to the demonstration region in both cases, we can see that the policies are capable of adapting effectively to the previously unseen scenarios.

D. Computational Advantage of Replication

Now we evaluate the potential benefit of exploiting the structure of replication during inference and prediction of the GP-based policy. From each of the 100 demonstrations we take 25 uniformly distributed timestamps for training. Thus, we have a total of $\mathcal{N} = 2500$ points with $n = 500$ unique input locations ($a = 5$). In Figure 9 we show a comparison of the computational time per evaluation of the inference function (inference), and for generating the task motion (prediction). We are able to perform the calculations 100 times faster without any approximation, achieving identical results.

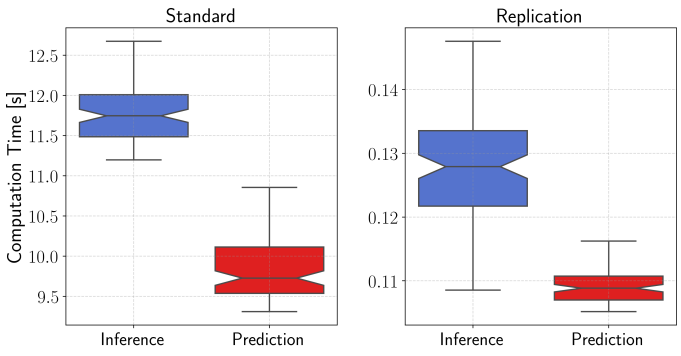


Fig. 9. Computational advantage of replication for evaluating the hyperparameter inference function, and calculating the predictive distribution.

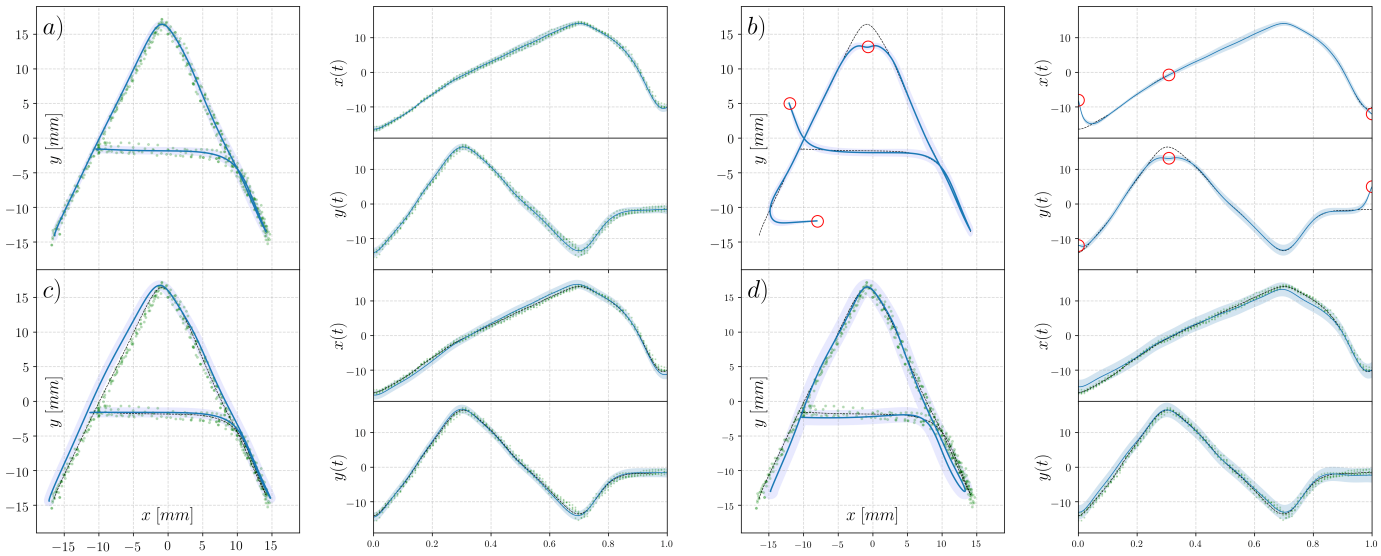


Fig. 8. **(a)** Motion retrieved by the task policy learned from the demonstrations for writing an 'A' of size 4. The solid blue line denotes the mean, the blue shaded area, the 95% confidence interval, and the green dots, the training data. **(b)** Policy adaptation through via-points (red circles). For comparison, the mean of the previous case is represented as a black dashed line. **(c)** The model is built only with demonstrations of sizes 3 and 5, thus interpolation is required to generate the required motion. **(d)** In this case, only demonstrations of sizes 5 and 6 are used, thus extrapolation is performed.

VI. Conclusion

Learning from Demonstration (LfD) is arising as a promising paradigm that allows intuitively transferring manipulation skills to robots. A central problem is to design a movement policy such that the retrieved motion can automatically adapt to new situations encountered by the robot. Also, currently, as the complexity of the taught manipulation tasks increases, there is a growing need for learning algorithms capable of handling large demonstrations datasets.

In this paper, we present a Gaussian-Process-based LfD framework that allows an expressive and versatile encoding of the policy. We focus on generalization performance and computational efficiency. Adaptability is enhanced by incorporating task variables into the model, which can be either real, integer or categorical. On the other hand, the scalability of the learning algorithm is boosted by exploiting the structure of replications, which arise naturally in the LfD context. The proposed approach is illustrated and tested by teaching the robot how to write, achieving satisfactory performance in terms of policy design, adaptability and computational savings. An important future challenge will be to extend the framework towards learning more complex tasks such as cloth manipulation.

References

[1] C.Nehaniv and K.Dautenhahn, "Like me?-Measures of correspondence and imitation," *Cybernetics and Systems*, vol. 32, pp. 11–51, 2001.

[2] P. Pastor, H. Hoffmann, T. Asfour, and S. Schaal, "Learning and generalization of motor skills by learning from demonstration," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2009, pp. 763–768.

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

[4] S. Calinon, "A tutorial on task-parameterized movement learning and retrieval," *Intelligent Service Robotics*, vol. 9, no. 1, p. 1–29, 2016.

[5] Y. Huang, L. Rozo, J. Silvério, and D.-G. Caldwell, "Non-parametric imitation learning of robot motor skills," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2019, pp. 5266–5272.

[6] J. A. Delgado-Guerrero, A. Colomé, and C. Torras, "Sample-efficient robot motion learning using Gaussian Process Latent Variable models," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2020, pp. 314–320.

[7] M. Arduengo, A. Colomé, J. Lobo-Prat, L. Sentis, and C. Torras, "Gaussian-process-based robot learning from demonstration," *arXiv e-print*, 2020, arXiv:2002.09979.

[8] Y. Huang, L. Rozo, J. Silvério, and D.-G. Caldwell, "Kernelized Movement Primitives," *The International Journal of Robotics Research*, vol. 38, no. 7, pp. 833–852, 2019.

[9] T. Matsubara, S. Hyon, and J. Morimoto, "Learning stylistic dynamic movement primitives from multiple demonstrations," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2010, pp. 1277–1283.

[10] D. Forte, A. Gams, J. Morimoto, and A. Ude, "On-line motion synthesis and adaptation using a trajectory database," *Robotics and Autonomous Systems*, vol. 60, no. 10, p. 1327–1339, 2012.

[11] E. Snelson and Z. Ghahramani, "Sparse Gaussian Processes using pseudo-inputs," in *18th International Conference on Neural Information Processing Systems (NIPS)*, 2005, p. 1257–1264.

[12] M. Schneider and W. Ertel, "Robot learning by demonstration with local Gaussian Process Regression," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2010, pp. 255–260.

[13] P. Senin, "Dynamic Time Warping algorithm review," Information and Computer Science Department, Univerity of Hawaii at Manoa, Honolulu, USA, Tech. Rep., Dec 2008.

[14] C.-E. Rasmussen and C.-K.-I. Williams, *Gaussian Processes for Machine Learning*. The MIT Press, 2006.

[15] E. Snelson and Z. Ghahramani, "Variable noise and dimensionality reduction for sparse Gaussian Processes," in *Proceedings of the Twenty-Second Conference on Uncertainty in Artificial Intelligence (UAI)*, 2006, pp. 461–468.

[16] M.-A. Alvarez, L. Rosasco, and N.-D. Lawrence, "Kernels for vector-valued functions: A review," *Foundations and Trends in Machine Learning*, vol. 4, no. 3, pp. 195–266, 2012.

[17] O. Roustant, E. Padonou, Y. Deville, A. Clément, G. Perrin, J. Giorla, and H. Wynn, "Group kernels for Gaussian Process Metamodels with categorical inputs," *SIAM/ASA Journal on Uncertainty Quantification*, vol. 8, no. 2, pp. 775–806, 2020.

[18] M. Binois, R.-B. Gramacy, and M. Ludkovski, "Practical Heteroscedastic Gaussian Process Modeling for large simulation experiments," *Journal of Computational and Graphical Statistics*, vol. 27, no. 4, pp. 808–821, 2018.

[19] S. Lemaignan, A. Jacq, D. Hood, F. Garcia, A. Paiva, and P. Dillenbourg, "Learning by teaching a robot: The case of handwriting," *IEEE Robotics Automation Magazine*, vol. 23, no. 2, pp. 56–66, 2016.