Learning Task Parameterised Dynamical Systems with Gaussian Process Regression and Classification

Mariano Ramírez Montero

Delft University of Technology

TU Delft

Learning Task Parameterised Dynamical Systems with Gaussian Process Regression and Classification

by

Mariano Ramírez Montero

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Thursday, December 14, 2023 at 3:00 PM.

Student number: 4682254
Thesis committee: Dr. Ir. J. Kober
G. Franzese
Dr. Ir. L. Peternel

TU Delft

Acknowledgments

The first people I want to thank are my parents, Mariano (the original Mariano) and Jeannette, and my sister Irene. They have supported me since I was a child interested in science, when I was eager to leave to far away places, and when it was taking me a bit longer than usual to figure things out. I would also like to thank my extended family: aunts, uncles, and cousins, for their unconditional love and support.

Getting here would not have been possible without great mentorship. My daily supervisor, Giovanni, provided great feedback and advice, and rekindled my enthusiasm for robotics. I would also like to thank my primary supervisor, Jens, for his feedback and attention, as well as Leandro for his help earlier in the process. Thanks also to all my past, supportive teachers, who are too many to mention.

The many days in the library would not have been tolerable without one of my best friends: Sara. Thank you for the many laughs, coffees, and vents. Thanks also to the many other friends who supported me: my best now-outside-the-TU-friends, Matthew (and Bebsi) and Serafeim, thanks for keeping me sane and grounded, and for all the night trains. To Joseph, thanks for all the talks, support, and memes. To Melissa for all the warmth and fun macumbas. To my fellow WAVEr's, Dorian, Eliana, Wiebke, Max, Khalid, and Maaike, and to WAVE in general for the community and all the training. To my robotics classmates and friends, Jeroen, Stan, and Cilia, for showing me that graduating is possible. To my far away UWC friends, Farah, Aunonna, and Anusha, and to the other UWCer's who I don't see often but remind me of great times, and to Omari and Jule for living close by, even if I never see you.

Finally, to all the random people and faces I became familiar with in that library but never met: good luck with your studies. I promise to make the very long journey from 3mE to visit sometimes.

Mariano Ramírez Montero
Delft, December 2023

Learning Task Parameterised Dynamical Systems with Gaussian Process Regression and Classification

Mariano Ramírez Montero

***Abstract*—Recent research has shown that a Learning from Demonstration (LfD) approach is useful for teaching robots flexible skills efficiently, and it opens the possibility for non-expert users to program these skills. When learning from demonstration data, learning frameworks should learn representations that are flexible and can generalize to unseen situations. Within the context of multi-reference frame skill learning, this work proposes a framework to learn such a representation without using task-specific heuristics or pre-segmentation of the demonstrations. Local policies are first learned by fitting the local dynamics with respect to each frame using Gaussian Processes (GP). A classifier that determines the relevance of each frame for every time step is then trained in a self-supervised manner. The uncertainty quantification capability of Gaussian Processes is exploited to improve the performance of the local policies and the self-supervised learning process of the classifier. The framework is validated through multi-frame tasks in simulation as well as on a robotic manipulator with a pick-and-place re-shelving task. Its performance is also compared to that of the Task-Parameterised Gaussian Mixture Model (TPGMM) with simulated and robotic data. In this comparison, the proposed model performs better according to metrics that quantify deviation from the goal at each reference frame and according to similarity measures between demonstrations and their corresponding reproductions.**

I. Introduction

As robots are becoming more ubiquitous in our society, it is necessary to easily provide flexible skills to them on the fly. A promising possibility is to use learning from demonstration to transfer knowledge and skills to the robot since this can allow easy programming of robots, even for non-roboticists. Additionally, to maximize the flexibility of the robot, their learned skills should generalize to new, unseen situations.

This work thus proposes a framework, Task Parameterised Gaussian Processes (TPGP), that can learn flexible skills that are parameterised by given coordinate reference frames. For example, the reference frame of an object that must be picked and the reference frame of a goal where it must be placed. If a human provides several continuous demonstrations of such a task, the framework is then able to learn a skill that achieves this pick-and-place task even when the object and goal are in new, unseen positions.

Apart from the given demonstrations and the respective coordinate frame positions for each of these demonstrations, the framework requires no other additional information as input. The framework learns to parameterise the skill in a self-supervised way, without using labels or manual segmentations of the provided demonstrations related to the coordinate frames. This makes the proposed framework more flexible and general than approaches that use task-specific heuristics to pre-segment or parameterise their method.

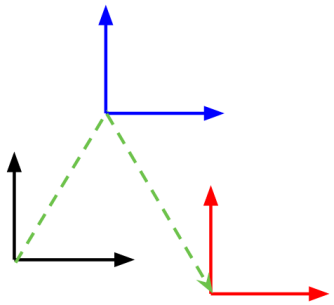


Fig. 1: Example task where first frame 1 (in blue) is approached, and then frame 2 (in red) is approached.

The structure of this report is as follows: Section II introduces the problem to be addressed. Section III summarizes the current related work, and Section IV explains the methodology of the proposed framework. Results in simulation are presented in Section V, while Section VI presents experiments performed on a robotic platform. Finally Section VII closes with conclusions and suggestions for future work.

II. Problem Formulation

We want to teach a robot how to perform tasks such as pick-and-place from human demonstration. To successfully perform the task the robot has to first follow the dynamics learned with respect to the object frame and then switch to the goal frame. A diagram of the task is depicted in Figure 1. In the most general case, we consider a task that involves m different reference frames defined with respect to a fixed frame, and we assume the location of these is observable. Then, given N demonstrations that include the position data of our agent (e.g. the end-effector Cartesian position of a robotic manipulator) and the positions of the potentially relevant reference frames as input, the goal is to design a framework that can generalize the task shown in these demonstrations when the m frames are in a new, unseen configuration.

This paper contributes to the topic of teaching multi-reference frame skills to perform pick-and-place tasks without using any explicit segmentation algorithm or heuristics for the reference frame selection. At every time step, "local" policies output dynamics relative to each frame while a classifier weighs the relevance of each of these. The proposed framework, based on Gaussian Process regression and classification, captures and rejects uncertainties in the learned local dynamics and ambiguities in the weighing of the frames' relevance.

III. Related work

When learning a multi-reference frame policy, there are usually two approaches. The first is to segment the trajectory into many sub-motions and then find the most relevant frame for each of them. Alternatively, rather than relying on a segmentation algorithm, other algorithms solve the allocation problem for each timestep in a continuous way.

A. Segmentation-based Generalization

In the context of object manipulation tasks, a common heuristic to segment demonstrations using changes in "contact relations." In practice, this usually means observing the distance between the end-effector and any relevant objects or the haptic sensory signals, and segmenting when these cross certain thresholds. This object-centered segmentation is exploited by these methods to make their methods more general. Since each segment is related to a given object, the learned skills can also be learned with respect to that object's position, so that the learned skill will generalize to unseen positions of the object.

For example, Wächter and Asfour [1] propose a two-layered segmentation where they first segment based on any changing contact relations, measuring these changes by thresholding the distance between the end-effector and other objects. These segments are further segmented using a custom criterion based on the acceleration profile.

Mühlig, Gienger, and Steil [2] also explore the idea of using contact relations to segment. Their demonstrations are provided directly by humans, instead of kinesthetically on the robot. They thus look at the relations between the demonstrator's hand and the manipulated objects. Different to the other methods that also use contact relations as a segmentation heuristic, they also consider the correlation between the object and the hand's velocity, which helps to avoid false positives when the hand passes close to an object but does not manipulate it.

Li, Li, Lu, *et al.* [3] also segment by thresholding the distance between the end-effector and objects in the scene. They use these segments to learn a two-level policy using hierarchical reinforcement learning. The low-level policies learn how to reach sub-goals while the high-level policy chooses which low-level policy to use given the current state, and both of these are trained in parallel.

Kober, Gienger, and Steil [4] similarly use contact relations changes to segment, but since their focus is force interaction tasks, they detect these changes by thresholding the measured force norm. Moreover, they also segment based on zero-velocity crossing (ZVC) events, i.e. when the magnitude of the velocity goes below a certain threshold, noting that they merge segmentation points that are very close to each other. They also note that using contact relations and ZVC has some biological basis, since these factors are also important for dexterous manipulation in humans [5]. Then, to assign reference frames to each segment, they use the idea of "convergence," where they assume that corresponding segments between different demonstrations should converge to the same goal. To measure this convergence, they look at the inter-trial variance of a segment in each candidate frame, where a decreasing variance as well as a low final variance indicates convergence and thus leads to a higher score for that frame.

Manschitz, Gienger, Kober, *et al.* [6] use a very similar approach, first segmenting based on ZVC, and also applying the concept of convergence. However, they formalize this concept further, proposing a novel Directional Normal Distribution (DND) which serves to quantify the similarity of segments using convergence. They thus assume that the ZVC heuristic will over-segment, and use DNDs to merge segments. In this merging process, the segments are also compared in all potential candidate frames, and thus the DND also provides a way to assign a goal and a reference frame to a group of segments, from which a motion primitive can then be learned.

One of the main disadvantages of using contact relations heuristics is that the segmentation method becomes specific to manipulation tasks. Moreover, most of these methods [1]–[4] involve empirically tuned thresholds, which also makes them less flexible, even for other tasks which are still manipulation-focused.

Other works like those of Pais, Umezawa, Nakamura, *et al.* [7] and Ureche, Umezawa, Nakamura, *et al.* [8] also exploit the variance in the data as a segmentation heuristic. They specify a criterion that looks at the difference between the intra-trial variance (variance in a given time window) and inter-trial variance (variance across demonstrations). The idea is that changes in a variable that occur systematically across demonstrations mark important constraints of the task. They thus consider the variables with the highest criterion as the most important and constraining variables. Additionally, by checking which candidate reference frame leads to the highest criterion, that frame can be assigned to a given time instance.

The methods mentioned previously result in skills that generalize well due to their segmentation approach. However, most of these methods make use of heuristics in their segmentation, which as mentioned before are task-specific. Moreover, these heuristics frequently require setting a segmentation threshold, which also hinders the flexibility of the method. The approach proposed in this work avoids the use of heuristics, instead training a classifier in a self-supervised manner that learns to combine learned local dynamical systems to imitate the demonstrated task.

B. Segmentation-free Generalization

The approach of Calinon, Alizadeh, and Caldwell [9] and Calinon [10], Task-parametrised Gaussian Mixture Models (TPGMM), does not explicitly segment demonstrations, but it can also learn skills from demonstration data that generalize to changing task parameters such as the reference frames of relevant objects. The approach first transforms the demonstrated trajectories to all potentially relevant frames and then encodes each of these using Gaussian Mixture Models (GMMs). Then, in a new configuration, each of these Gaussians can be linearly transformed according to the new position of their corresponding frame. The resulting GMM for this situation can

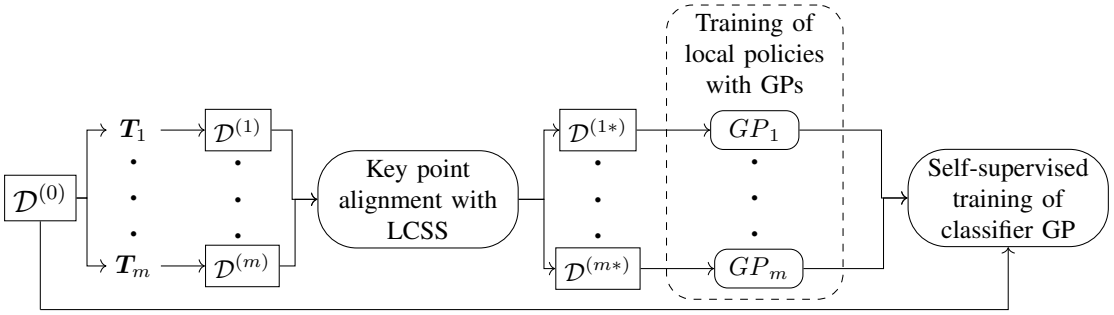


Fig. 2: TPGP framework pipeline, with the main sub-steps of the process indicated with rounded corner rectangles, and inputs and outputs indicated with regular rectangles.

then be computed as the product of the transformed models, exploiting the fact that the product of two Gaussians is still a Gaussian. This framework is also interesting since it is not limited to the task parameters being coordinate frames, instead, these could be other locally linear transformations or projections.

TPGMM is currently one of the state-of-the-art methods to address task parametrization in problems such as the one described in Section II. The proposed TPGP is thus compared with TPGMM in a two frame and a three frame task in simulation and using robotic data from a re-shelving task, and in both cases, the proposed method shows better performance in the majority of the proposed metrics.

IV. Methodology

To more easily explain the methodology, a specific scenario will be used as a running example throughout the rest of this work. We consider a task in 2D Cartesian space where the agent starts at (0,0) of the fixed frame. Two frames are defined relative to the origin of this fixed frame, and the task is to first go to the origin of frame 1, and then go to the origin of frame 2, as shown in Figure 1.

The main idea of the proposed approach is to transform the demonstration data to the local reference frames to encode the relative dynamics, and then use a self-supervised approach to train a classifier that selects the most relevant frame during execution. A diagram showing the main steps of the proposed framework is shown in Figure 2. As the diagram shows, the main steps are:

1) Alignment of the demonstrations using the longest common subsequence (LCSS) algorithm [11] to find alignment key points.
2) Training of local policies using Gaussian Processes (GPs).
3) Self-supervised training of a Gaussian Process classifier which learns the relevance of each frame.

Each of these steps is explained in detail in Section IV-B, Section IV-C, and Section IV-D, respectively.

A. Demonstration Recording

Before explaining the main steps of the proposed method, the input data that is recorded will be explained in this section. The notation used throughout this work is also introduced.

During the recording of a demonstration, the 2D/3D position of the agent, e.g. the robot, is recorded at a given frequency. As IV-C will show, time information is also required for this task. Each recorded position is thus augmented with a progress value φ between 0 and 1, which is set during the alignment step, after resampling, see Subsection IV-B.

The state $\boldsymbol{x}$ of the system is thus composed of the Cartesian position (2D or 3D) plus the progress value φ. A set of sequential states $\boldsymbol{x}$ with final time index t_f thus defines a demonstration $d_n = \{\boldsymbol{x}_0, \dots, \boldsymbol{x}_{t_f}\}$, where the subscript n denotes the n-th demonstration. The set of all given training demonstrations is then $\mathcal{D}^{(0)} = \{d_0^{(0)}, \dots, d_N^{(0)}\}$, where we add the superscript to indicate this is in the fixed, global (0th) frame, and N is the total number of demonstrations.

Moreover, before each demonstration, the positions of any relevant frames (relative to the fixed frame) are also recorded. A homogeneous transformation matrix $\boldsymbol{T}_m$ can then be defined for the m-th frame. This results in another m transformed demonstration sets $\mathcal{D}^{(1)}, \dots, \mathcal{D}^{(m)}$, as also shown in Figure 2, where again the (m) superscript is used to indicate the associated frame; this notation is used throughout the text for other variables. These sets, plus the original set in the fixed frame, are then the input to the proposed framework. The final output is then a policy that takes as input the current positions relative to each relevant frame, plus the current progress value φ, and outputs a displacement in space.

B. Alignment of demonstrations

As IV-C will discuss in more detail, the time information in the demonstrations is necessary to train the local policies. This is why the first step is aligning the demonstrations, since imperfect (human) demonstrations and the changing frame configurations will cause the demonstrations to be misaligned. The most common method to perform alignment, both in learning from demonstration as well as in the broader time series literature [12], is dynamic time warping (DTW) [13], used in [4], [14] in a robotics context, for example. Dynamic

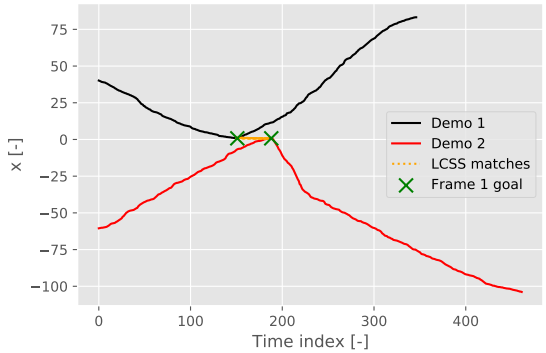


Fig. 3: Two example demonstrations transformed to frame 1. The matches found using the LCSS algorithm are shown with yellow lines. Note how several points close to the annotated goals are matched.

time warping finds an optimal alignment path between two time series using the pairwise distance between each of their points.

However, in this case, it is not straightforward to apply DTW to find a global alignment between all the demonstrations, and/or in all the different frames of reference. The reader is referred to Appendix A for a more detailed explanation which shows why DTW fails in this case, even if different pairwise alignments could be combined to find a global alignment. Thus instead of DTW, this work proposes to use the longest common subsequence algorithm (LCSS) [11] to find key points that should be aligned among all the time series.

The LCSS algorithm finds the longest subsequence between two time series that meets a certain similarity threshold and appears in the same order, but does not require the elements of the subsequence to be contiguous. The detailed algorithm is included in Appendix A. For example, if the algorithm was applied to the sequence "ABHDE" and "ZBCADE", the algorithm would return the longest common subsequence as "BDE" with length 3, and the path of matched indices (1,1), (3,4), (4,5). Points in each sequence are matched if their similarity meets a threshold. In the case of the demonstrations, for example, Euclidean distance is used as a distance measure (or an inverse similarity measure), and a maximum threshold for points to be matched is set.

Figure 3 shows examples of matches found between two demonstrations (transformed to frame 1) for the two frame tasks using the LCSS algorithm, where the "goals" in each of the demonstrations have been manually annotated as the points closest to the origins of the relevant frame.

The LCSS algorithm is used to find the matches for each demonstration in a given transformed dataset $\mathcal{D}^m$ with every other demonstration in that dataset. The index that appears the maximum amount of times in the LCSS paths is then kept as an alignment point for that demonstration and frame m, and if there are ties the last index is kept. After applying this process to all transformed demonstrations, the demonstrations in the global fixed frame can be aligned using these points by resampling.

C. Training the local policies

The first step in training the local policies is deciding what the training inputs and outputs should be. Regarding the output, the local policies are encoded with a Gaussian Process (GP) [15] as a dynamical system, thus outputting a displacement $\Delta \boldsymbol{s}$ given the state input $\boldsymbol{x}$.

$$\Delta \boldsymbol{s} = f(\boldsymbol{x}) \tag{1}$$

As will be shown in this section, the state is encoded with position plus a progress state similar to time, whereas the output from the GP is a displacement in space. This is why the output here is denoted with a different variable, s. The choice of state and how the progress variable increment is computed (see (11)) are explained below.

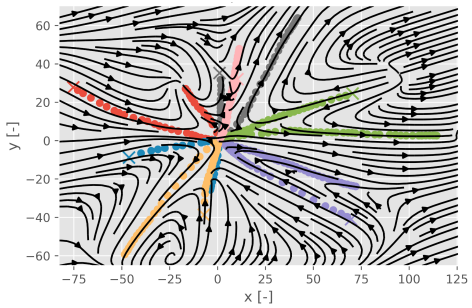
One could choose to encode the input state $\boldsymbol{x}$ using position, time, a progress scalar, or a combination of them. Using only position could be advantageous since it means time misalignment of the demonstrations is not a problem. However, this introduces the limitation that the demonstrated trajectories should not overlap in space, as this will introduce ambiguity in the training labels. To solve the example problem described at the beginning of this section or any similar problems, it is impossible for the given demonstrations to not overlap, thus this limitation is not acceptable. Figure 4a shows how a policy encoded with only space information can result in trajectories that miss the goal.

Similarly, encoding the behavior based only on a time or progress variable does not work in the context of this task. A specific, constant displacement will always be output for a given time regardless of the current position, which means that again the goal will be missed depending on the initial position. This behavior can also be observed in Figure 4b. However, if both position and time are included in the state $\boldsymbol{x}$, we finally obtain a dynamical system that results in trajectories consistently going towards the goal, as can be seen in the stream plot in 4c. Finally, note that instead of a time value, a progress variable φ is used which is simply the time normalized by the total time length of each demonstration, meaning $0 \leq \varphi \leq 1$, and note that this is computed after alignment.

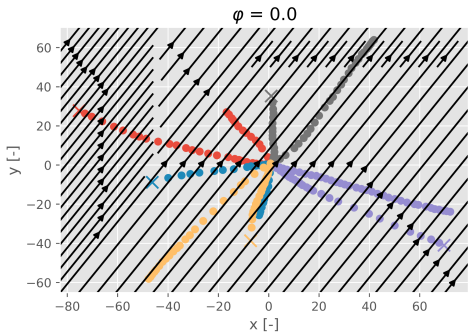
As mentioned before, the local policies are encoded using Gaussian Processes (GPs). A GP is a non-parametric regression method where prior mean and kernel functions are first specified. Then, given some observed data, Bayes's theorem is applied to compute a posterior distribution after updating the prior distribution given the likelihood of the recorded data. At execution time, the posterior distribution is used to infer new outputs by conditioning it on new test points [15], obtaining a mean μ and a variance Σ,

$$\mu(\boldsymbol{x}) = \boldsymbol{k}_*(\boldsymbol{X}, \boldsymbol{x})^\top (\boldsymbol{K}(\boldsymbol{X}, \boldsymbol{X} + \sigma_n^2 \boldsymbol{I}))^{-1} \boldsymbol{y} \tag{2}$$

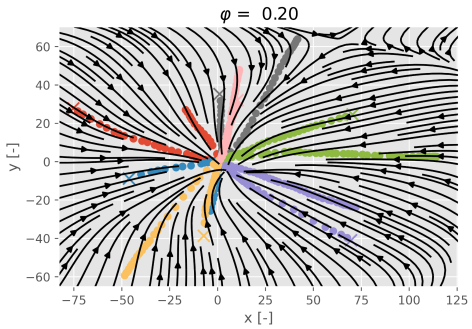
$$\Sigma(\boldsymbol{x}) = k(\boldsymbol{x}, \boldsymbol{x}) - \boldsymbol{k}_*(\boldsymbol{X}, \boldsymbol{x})^\top (\boldsymbol{K}(\boldsymbol{X}, \boldsymbol{X} + \sigma_n^2 \boldsymbol{I}))^{-1} \boldsymbol{k}_*(\boldsymbol{X}, \boldsymbol{x}), \tag{3}$$



(a) Stream plot with position-only encoded state.



(b) Stream plot with time-only encoded state.



(c) Stream plot with position and time encoded state.

Fig. 4: Stream plots of resulting dynamical systems for different state encodings. Different training demonstrations are shown in different colors.

where $\boldsymbol{k}_*$ is the covariance between the training inputs $\boldsymbol{X}$ and test point $\boldsymbol{x}$, $\boldsymbol{K}$ the covariance matrix of the training inputs $\boldsymbol{X}$, k is the variance of the test point x, and σ_n^2 is the noise variance; $\boldsymbol{X}$ and $\boldsymbol{y}$ are the set of observed training inputs and training outputs. The terms $\boldsymbol{k}_*$, $\boldsymbol{K}$, and k are all functions of the chosen prior kernel function and its hyperparameters, which can also be learned instead of chosen with the prior.

Equations (2) and (3), however, apply to an exact GP. One disadvantage of these is that due to the inversion of the covariance matrix $\boldsymbol{K}$, training becomes computationally expensive as the training dataset grows. Thus, as an alternative, a variational approximation of the posterior is used for this framework instead. A Stochastic Variational Gaussian Process (SVGP) approximates the posterior distribution using a variational distribution [16] parameterised by inducing points $\boldsymbol{Z}$ (smaller in number than the full training dataset) and inducing values $\boldsymbol{u}$, both of which are learned from the data. The predictive mean and variance then become, respectively,

$$\mu(\boldsymbol{x}) = \boldsymbol{k}_*(\boldsymbol{Z},\boldsymbol{x})^\top (\boldsymbol{K}(\boldsymbol{Z},\boldsymbol{Z}+\sigma_n^2\boldsymbol{I}))^{-1}\boldsymbol{u} \tag{4}$$

$$\Sigma(\boldsymbol{x}) = k(\boldsymbol{x},\boldsymbol{x}) - \boldsymbol{k}_*(\boldsymbol{Z},\boldsymbol{x})^\top (\boldsymbol{K}(\boldsymbol{Z},\boldsymbol{Z}+\sigma_n^2\boldsymbol{I}))^{-1}\boldsymbol{k}_*(\boldsymbol{Z},\boldsymbol{x}). \tag{5}$$

The variational parameters are obtained by maximising the Expected Lower Bound (ELBO) of the likelihood of the observed labels given the approximated posterior.

One of the main advantages of using GPs is their epistemic uncertainty quantification through the predictive variance. As will be shown later in this subsection, within the context of learning a dynamical system, this uncertainty can be used to guide the system away from uncertain regions of the input space, making it more likely that the goals will be reached. Moreover, as will be shown in Section IV-D, the uncertainty of each of the local GPs can also help the classifier disambiguate between frames by choosing the local GP that will lead to lower uncertainty.

One local policy can then be trained for each frame using the transformed demonstration sets. The position in Cartesian space plus the progress variable φ of each demonstration becomes the training input $\boldsymbol{X}$, and the displacements at each step of each demonstration become the training labels $\boldsymbol{y}$. For our running example in 2D Cartesian space, we would then have a dynamical system of the following form for each frame:

$$\begin{bmatrix}\Delta x_i^{(m)} \\ \Delta y_i^{(m)}\end{bmatrix} = \Delta \boldsymbol{s}_i^{(m)} = \boldsymbol{\mu}_m(\boldsymbol{x}_i^{(m)}) \tag{6}$$

$$\boldsymbol{x}_{i+1}^{(m)} = \boldsymbol{x}_i^{(m)} + \begin{bmatrix}\Delta x_i^{(m)} \\ \Delta y_i^{(m)} \\ \Delta \varphi_i\end{bmatrix} \tag{7}$$

Additionally, as mentioned earlier, the uncertainty of the GPs can be used to pull the system towards regions of higher certainty. Specifically, we can take the derivative of (3) with respect to the inputs,

$$\nabla\boldsymbol{\Sigma} = \begin{bmatrix}\frac{\partial\Sigma}{\partial x} \\ \frac{\partial\Sigma}{\partial y} \\ \frac{\partial\Sigma}{\partial \varphi}\end{bmatrix} = 2\boldsymbol{k}_*^\top(\boldsymbol{K}+\sigma_n^2\boldsymbol{I})^{-1}\frac{\partial \boldsymbol{k}_*}{\partial \boldsymbol{x}} \tag{8}$$

and use the direction of this gradient to pull the system towards regions of higher certainty, minimizing the variance. The resulting vector is then scaled by the standard deviation so that the correcting effect is stronger the higher the uncertainty. Accordingly,

$$\nabla\boldsymbol{\Sigma}_{x,y}^{(m)} = \begin{bmatrix}\frac{\partial\Sigma}{\partial x} \\ \frac{\partial\Sigma}{\partial y}\end{bmatrix} \tag{9}$$

$$\boldsymbol{v}_i^{(m)}(\boldsymbol{x}_i^{(m)}) = p_v\Sigma^{1/2}(\boldsymbol{x}_i^{(m)})\frac{\nabla\boldsymbol{\Sigma}_{x,y}^{(m)}}{\|\nabla\boldsymbol{\Sigma}_{x,y}^{(m)}\|} \tag{10}$$

where the constant scalar p_v is added as a gain for the variance minimization term. This variance minimization term can then be expressed for each dynamical system as shown in (10), denoted as v_i^m for a frame m and a given step i.

The authors of [17] and [18] showed how the variance minimization term is beneficial, since it helps the robotic manipulator reject disturbances that might otherwise lead the robot to areas of high uncertainty, ultimately causing the robot to fail in performing reaching task due out-of-distribution compounding errors.

Finally, it is still necessary to determine $\Delta\varphi$, the increment for the progress variable. In situations similar to those in the training demonstrations, a constant increment might work as long as it is close to the average increment in the demonstrations, but this will not always generalize well. Instead, the covariance function of the Gaussian process can be exploited to dynamically adjust the phase increment. First, the correlations of the current state $\boldsymbol{x}_i$ with all the training points X are calculated. Then, we can look at the progress values φ for the points with the highest correlation (e.g. top 10), and calculate an average, denoted as φ_{corr}. The difference between this average φ_{corr} and the current φ_i can be used to increase or decrease the rate of the progress variable φ.

Thus, the progress increment $\Delta\varphi$ at each time step becomes

$$\Delta\varphi_i(\boldsymbol{x}_i) = c_\varphi + p_\varphi c_\varphi(\varphi_{corr}(\boldsymbol{x}_i) - \varphi_i) \tag{11}$$

with c_φ being a constant increment, and p_φ working as a proportional "gain" for this additional term. Intuitively, in our running example, this would mean that the progress φ would increase faster if the initial goal is close to the starting position, and vice-versa, since the average progress value at which the goal is reached is likely close to 0.5. Note that this strategy would not work as well if the demonstrations were not aligned, since φ would be less consistent for a given set of highly correlated points. The authors of the 'SIMPLe' framework [19] show how such a strategy results in safer policies for bimanual robotic tasks. Moreover, they also show how considering the time or progress along with the position is especially important to avoid ambiguities in overlapping demonstrations.

D. Self-Supervised Training of the Classifier GP

The final component of this framework is the classifier, which determines the relevance of each frame at every timestep. Another Stochastic Variational Gaussian Process (SVGP) with a softmax likelihood is used as the classifier.

Similarly to the local GPs, the input of the classifier must first be chosen. Using the running example of the two frame task (see Figure 1), it is possible to show that the relative positions to each of the frames are not always informative regarding frame relevance, so using these as the input could produce ambiguities. The easiest way to visualize this issue is to consider the case where the starting point and the two frames are co-linear, but the first frame is further than the second. On the line between the two frames the demonstration would overlap, and these positions would be associated with frame 1 (when going towards frame 1) and at the same time to frame 2 (when coming back to frame 2). On the other hand, if we use the progress variable φ as input to the classifier, this ambiguity does not occur, since no overlap is possible due to the uniqueness (within a given demonstration) of this value for each datapoint. Moreover, the initial alignment step which helped to mitigate ambiguities during the training of the local GPs will do the same in the classifier training.

As explained in the introduction, one of the goals of this framework is to learn such a task without having to generate labels for the frame relevance. First, the local policies are trained as described in IV-C with a subset of the total demonstrations. Then, a different, non-overlapping subset is used as input to get predictions from all the trained local GPs. A loss can then be designed which enables the self-supervised training of this classifier.

The predictions from the trained local GP policies are transformed to the global fixed frame using the inverse transform $\boldsymbol{T}_m^{-1}$ for a frame m, and compared to the displacement labels of the input demonstrations that were used to produce predictions in the first place. The classifier's loss function can then be based on picking the local policy whose predictions are most similar to the labels. To quantify similarity, the cosine similarity is used, which is defined according to (12) for two vectors $\boldsymbol{a}_1$ and $\boldsymbol{a}_2$. It is the dot product of two vectors normalized by their magnitude, which means it will range between 1 for perfectly aligned vectors to -1 for vectors of opposite directions,

$$C_s(\boldsymbol{a}_1, \boldsymbol{a}_2) = \frac{\boldsymbol{a}_1 \cdot \boldsymbol{a}_2}{\|\boldsymbol{a}_1\| \cdot \|\boldsymbol{a}_2\|}. \tag{12}$$

Now if we consider a classifier that outputs a vector $\boldsymbol{\alpha}$ of probabilities or weights (one per frame), which is softmaxed so that these add up to 1, a loss function for a total of m_t frames that uses this similarity function would be:

$$J_s = \sum_{m=1}^{m_t} -C_s(\boldsymbol{T}_m^{-1}\Delta\boldsymbol{s}^{(m)}, \boldsymbol{y}^{(0)})\boldsymbol{\alpha}^{(m)}. \tag{13}$$

Note that the superscript (m) here indicates the associated frame, and is not an exponent. The negative signs are needed for the minimization since the cosine similarity outputs $+1$ for maximum similarity. Minimizing this cost function would then encourage high relevance weights $\boldsymbol{\alpha}$ for the local GP policy that produces the most similar predictions to the global labels.

Using only the similarity in the loss, however, leads to ambiguities for the classifier. Approximately during the second half of the progress values (0.5 to 1.0), the first local GP has learned a "divergent" behavior while the second local GP has learned an "attractive" behavior. After reaching the first goal (the origin of frame 1), when going between frames 1 and 2, we are evaluating the predictions of the local GPs along a line between the two frames, since this is what was demonstrated. Thus, both local GPs, with their divergent and convergent behaviors, will predict a displacement that has a similar direction to the label when they are evaluated along this line, resulting in ambiguity.

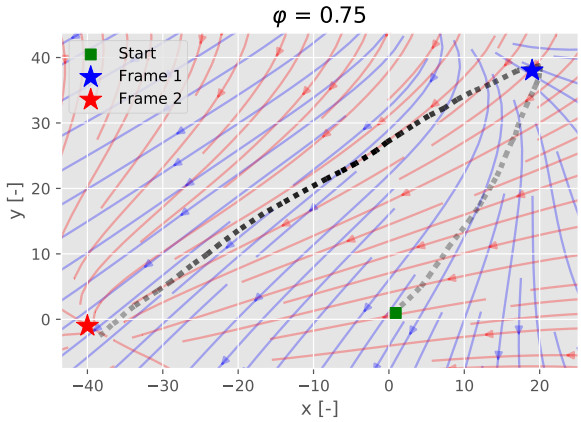


Fig. 5: Learned dynamics of local GPs for running example at a progress value of 0.75, shown in blue and red streamlines for frames 1 and 2, respectively. An example configuration of the frames is shown with the stars, and the provided demonstration is shown as a dotted line. Note how along the highlighted area of the line, the streamlines from both local GPs strongly align.

This ambiguity issue is depicted in Figure 5, where an example configuration of frames 1 and 2 is shown with the blue and red stars, respectively, and the demonstration given for this configuration is shown as the black dashed line. Both learned dynamical systems are shown as blue and red streamlines, for frames 1 and 2 respectively, at a progress value $\varphi = 0.75$. The section of the demonstration also corresponding to this progress value is highlighted with darker black. Along the highlighted area of the line, it is possible to see how the blue and red streamlines strongly align, which leads to the mentioned ambiguity.

To attempt to resolve this ambiguity, an additional term that takes advantage of the predictive uncertainty of the local GPs is used. This additional term is called $\boldsymbol{g}_m$, and to quantify it first the variances for the training inputs corresponding to each of the demonstrations are calculated using each of the trained local GPs. Then, the local change in the variance at each timestep for every training demonstration can be computed by taking the difference between the variance at step $i+1$ and the variance at step i. This is calculated for every local GP belonging to a given frame m. In this case, we will look only at the direction, i.e. whether the resulting difference is negative or positive.

If it is negative, the local GP's own variance is decreasing by following its prediction, otherwise, it is increasing. Thus by looking at the sign of this change, we can check whether the direction of the prediction of each local GP aligns with the direction that will decrease its variance. This $\boldsymbol{g}_m$ term can then be used in the loss to encourage the classifier to learn to choose the local GP whose output will be most aligned with the direction that will minimize its variance.

For a total amount m_t of frames, this additional loss term can be computed as:

$$J_g = \begin{cases} \sum_{m=1}^{m_t} \boldsymbol{\alpha}^{(m)} \boldsymbol{g}^{(m)}, & \text{if } C_s(\boldsymbol{T}_m^{-1} \Delta \boldsymbol{s}^{(m)}, \boldsymbol{y}^{(0)}) > 0 \\ 0, & \text{otherwise} \end{cases}. \tag{14}$$

Additionally, note the conditional nature of this term: it is only considered for a given frame m if the similarity of the respective local policy prediction is similar to the label $\boldsymbol{y}^{(0)}$ (measured by the cosine similarity being greater than 0), otherwise it is 0. This ensures that this additional term is only considered when the prediction of the m-th frame is similar to the labels, since the goal of this additional term is to disambiguate in cases where more than one frame is similar to the labels. An ablation experiment in Section V-B shows how this term improves the output of the classifier, see Figures 10a and 10b.

The final loss J to be minimized is then simply the sum of J_s and J_g,

$$J = J_s + \lambda_g J_g, \tag{15}$$

with an added weighing term λ_g for the additional variance term J_g.

E. Executing learned tasks

With a trained classifier and the trained local policies, the whole framework can now be put together. For a given timestep i, a prediction for the weights $\boldsymbol{\alpha}$ is made by the classifier:

$$\begin{bmatrix} \alpha^{(1)} \\ \vdots \\ \alpha^{(m)} \end{bmatrix} = \boldsymbol{\mu}_{class}(\varphi_i). \tag{16}$$

Each of the predictions from each of the m local GPs, plus their variance minimization term, are multiplied by their respective weight α_m to get the global displacements:

$$\begin{bmatrix} \Delta x_i \\ \Delta y_i \end{bmatrix} = \sum_{m=1}^{m_t} \alpha_i^{(m)} \begin{bmatrix} \Delta x_i^{(m)} \\ \Delta y_i^{(m)} \end{bmatrix} + \alpha_i^{(m)} \boldsymbol{v}_i^{(m)}. \tag{17}$$

The progress increment is then calculated,

$$\Delta \varphi_i = c_\varphi + \sum_{m=1}^{m_t} \alpha^{(m)} p_\varphi c_\varphi (\varphi_{corr}^{(m)} - \varphi_i) \tag{18}$$

where the difference between the current progress value φ_i and the term φ_{corr}^m is also multiplied by the respective weight α_m. Finally, the next state $\boldsymbol{x}_{i+1}$ is calculated using the previous results:

$$\boldsymbol{x}_{i+1} = \boldsymbol{x}_i + \begin{bmatrix} \Delta x_i \\ \Delta y_i \\ \Delta \varphi_i \end{bmatrix}. \tag{19}$$

V. Experimental simulation results

As explained in Section IV-A, during each demonstration the agent's position is recorded at a regular interval, along with the initial position of any relevant coordinate frames. For the simulation experiments, this is done through a (2D) "drawing" interface where the demonstrations are given using a mouse. Throughout this section, the average of the minimum distances to each of the goals (e.g. the origins of each frame for the two frame task) in the generated reproductions is used as a performance metric. The average Fréchet distance [20] between the generated trajectories and the given demonstrations is also used as a performance metric.

The results are shown for the training configurations as well as for held-out test configurations. For all of these metrics, lower values indicate better performance. When comparing the proposed model to other models, the Mann-Whitney U test [21] is used to check whether the results from which the average metrics are calculated differ significantly. This helps to ensure that a lower calculated average (thus better performance) is highly unlikely to be the result of randomness. A threshold p-value of 0.05 is used, and metrics where the U test succeeded are highlighted in bold in the tables.

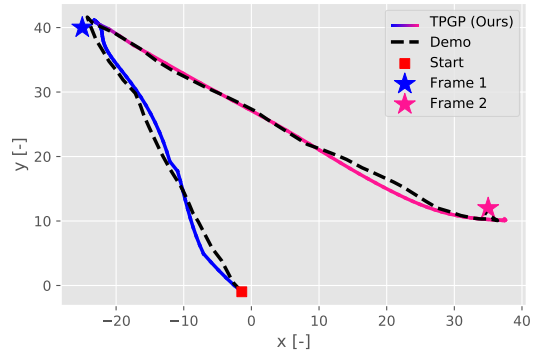
Results on the two frame problem described at the beginning of Section IV and shown in Figure 1 are given in Section V-A. Results on an additional writing task are also provided to show the flexibility of the framework. Section V-B shows ablations of the framework to highlight the use of variance minimization in the local policies and the additional variance gradient term in the loss of the classifier. Finally V-C compares the performance of the proposed framework, TPGP, with TPGMM on the two frame problem, and an extension with three frames.

A. *Two frame task experiments*

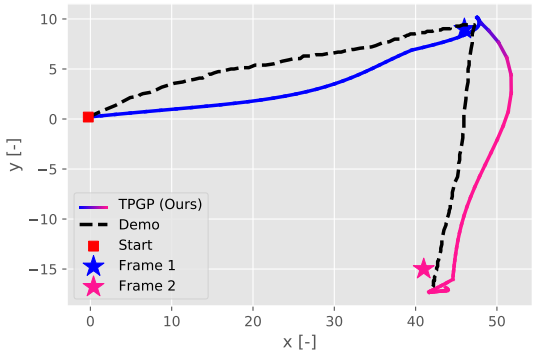
The proposed TPGP is trained on 10 demonstrations for the two frame task described at the beginning of Section IV and shown in Figure 1, where the origin of frame 1 is first approached and then the origin of frame 2 is approached. Generated trajectories for a training configuration, as well as for the held-out test configuration of the reference frames, are shown in Figure 6. In both cases the generated trajectories qualitatively show the desired behavior, going towards frame 1 and then towards frame 2. (More detailed numerical performance metrics for this task are provided in Section V-C when comparing to TPGMM.)

It is also desirable to test whether the model can ignore frames that are irrelevant to the task. A TPGP model is thus retrained on this same task, but a third frame (which is not approached or used in any of the demonstrations) is added, i.e. a third local policy is also trained with the data relative to this frame. Ideally, the classifier would then output a relevance weight of 0 for this third frame. Figure 7 shows the classifier output for an example training configuration of the frames, where we indeed see that this third frame is ignored.

Finally, the proposed model is also tested on a writing task that involves two frames to highlight its flexibility. In this task, the agent has to approach the first frame in the same way but



(a) Generated trajectory for a training configuration. The demonstration is shown as the black dashed line.



(b) Generated trajectory for a randomly generated test configuration.

Fig. 6: Generated trajectories for the two frame task. The color of the trajectories visualizes the relevance weight output by the classifier, with blue corresponding to frame 1 and pink to frame 2.

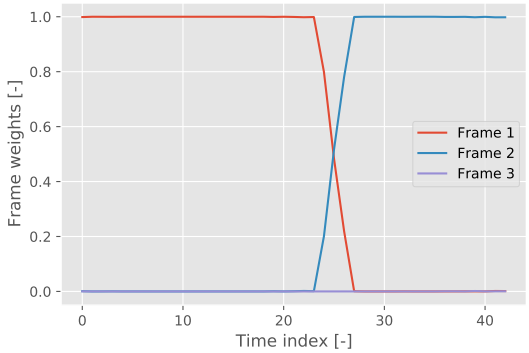
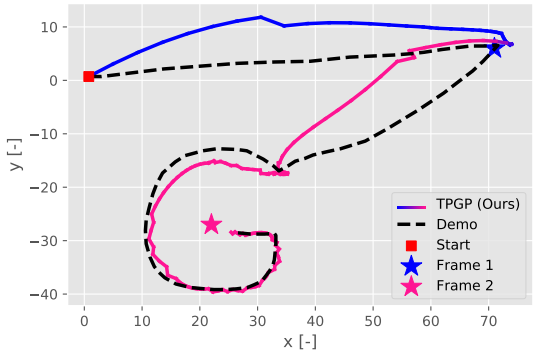
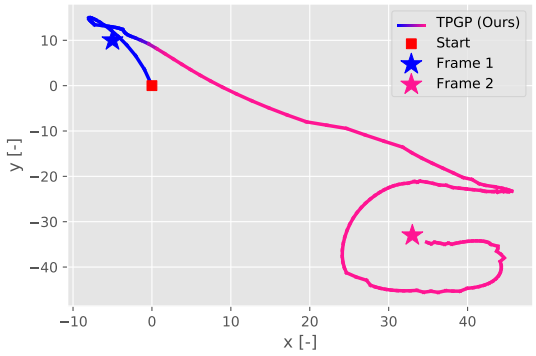


Fig. 7: Relevance weights learned by the classifier for the running example task after a third, irrelevant frame is added.



(a) Generated trajectory for a training configuration. The demonstration is shown as the black dashed line.



(b) Generated trajectory for a randomly generated test configuration.

Fig. 8: Generated trajectories the writing task. The color of the trajectories visualizes the relevance weight output by the classifier, with blue corresponding to frame 1 and pink to frame 2.

then has to draw the uppercase letter 'G' centered on the origin of the second frame. Figure 8 shows generated trajectories for a training and test configuration. Qualitatively, the model again shows the expected performance, reaching the origin of frame 1 and then tracing the letter, even though it does deviate from the demonstration when approaching these goals.

B. Ablation experiments

Ablation experiments are also performed to highlight the importance of certain elements. Specifically, the model is tested without using any minimization of variance for the local policies, and the classifier is tested without the additional variance term in its loss.

The full model is tested on the 10 training configurations and 5 test configurations for the two frame task, where everything is identical to the earlier experiments except the variance minimization gain, i.e. p_v in (10), is set to 0. Table I summarizes the results and highlights how the model with variance minimization performs better in almost every metric.

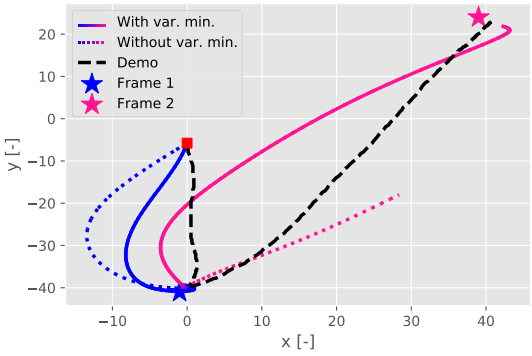


Fig. 9: Generated trajectories by models with and without variance minimization for a test configuration of the two frame task.

TABLE I: Performance metrics for TPGP with and without variance minimization. For all metrics, lower values indicate better performance. Bold values indicate that the values used to calculate the average were significantly lower according to the Mann-Whitney U test.

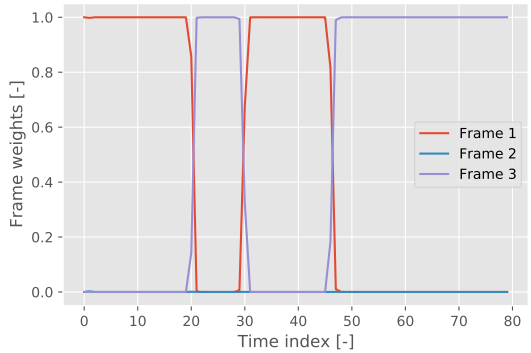
	Average distance to goal 1 [-]		Average distance to goal 2 [-]		Average Fréchet distance [-]	
	train	test	train	test	train	test
TPGP w/o var. min.	0.64	0.68	3.03	7.36	5.93	10.08
TPGP with var. min.	**0.18**	**0.25**	**0.59**	**0.58**	**4.76**	14.12

Note also how in the case where the model with variance minimization performed worse (average Fréchet distance for the test case), the difference in the results was not significant according to the Mann-Whitney U test.

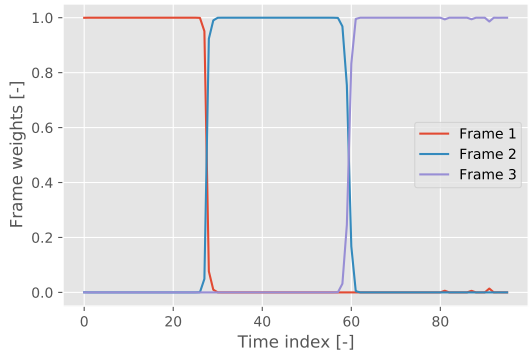
A trajectory plot of one of the test configurations is shown in Figure 9. The trajectory generated by the original model is shown with a solid line while the model without variance minimization uses a dotted line. This also visualizes how the variance minimization helps the model achieve the demonstrated task, especially when the system is in regions of the input space that are far from the training data.

Similarly, the classifier loss (see (15)) is tested without the additional variance term J_g (see (14)) by setting its weight λ_g to 0. The full model is trained on 10 demonstrations for the three frame task, where first frame 1 is approached, then frame 2, and finally frame 3. The output from the classifier should show a sequence where it is initially high for frame 1, then for frame 2, and finally for frame 3.

Figure 10a shows the classifier output when this J_g term is removed from the loss. While the classifier can correctly classify the first and last sections of the demonstration to frame 1 and frame 3, it fails to classify the middle section to frame 2. Using this classifier would lead to the trajectory missing the second goal, and potentially diverging to unknown regions



(a) Output weights for a model trained without the additional variance term J_g.



(b) Output weights for a model trained with the additional variance term J_g.

Fig. 10: Classifier weights for the three frame task with and without the additional variance term J_g in the loss. Note how the expected relevance weights are output by the model when this term is added, while the model without this term fails to output a high relevance for frame 2 in the middle of the trajectory.

of the state space. This misclassification is likely due to the ambiguity explained in Section IV-D and visualized in Figure 5. Once the additional penalty term is added, the classification shows the expected behavior, as shown in Figure 10b.

C. Comparison with TPGMM

To evaluate the proposed method, its performance is compared to that of the state-of-the-art TPGMM algorithm [10], [22]. As mentioned in Section III, TPGMM fits GMMs on the data transformed to each of the relevant frames. Then at execution time, these can be transformed to the new configuration of the frames, and multiplied together to find the new GMM for this configuration. Specifically, an implementation of TPGMM is used where the resulting GMM is considered as an approximation of a Hidden Markov Model (HMM). At execution time, the Viterbi algorithm is used to determine the most likely sequence of hidden states from a training demonstration,

TABLE II: Performance metrics for TPGP and TPGMM-LQR for the two frame task. For all metrics, lower values indicate better performance. Bold values indicate that the values used to calculate the average were significantly lower according to the Mann-Whitney U test.

	Average distance to goal 1 [-]		Average distance to goal 2 [-]		Average Fréchet distance [-]	
	train	test	train	test	train	test
TPGMM-LQR	1.96	0.85	0.48	0.66	9.95	9.44
TPGP (Ours)	**0.18**	**0.49**	**0.25**	**0.25**	**4.26**	13.64

where each of these states corresponds to one of the Gaussian components of the GMM, and a linear-quadratic regulator (LQR) is employed to track the generated trajectories. This specific version of TPGMM is called TPGMM-LQR[1] in the following experiments.

The performance of TPGP and TPGMM-LQR is compared for two cases: the example problem from the beginning of Section IV, and an extension of that problem where a third frame has to be reached after the second frame. Their performance is tested on the 10 training demonstrations and 5 new, held-out test frame configurations.

For the two frame task, Table II shows the results. In both the training and test frame configurations, the proposed model performs better than TPGMM-LQR in more metrics. Specifically, note that for the distance to goal metrics, the TPGP results are about half those of TPGMM-LQR.

Examples of generated trajectories by the proposed TPGP model and TPGMM-LQR for the two frame task are shown in Figures 11a and 11b for a training and test configuration, respectively. Qualitatively, both models seem to perform similarly in terms of reaching the goals, though note that as shown in Table II, TPGP does deviate less from the goals. Figure 11a is also an example where TPGMM-LQR deviates from the straight-line path to the first goal more than TPGP.

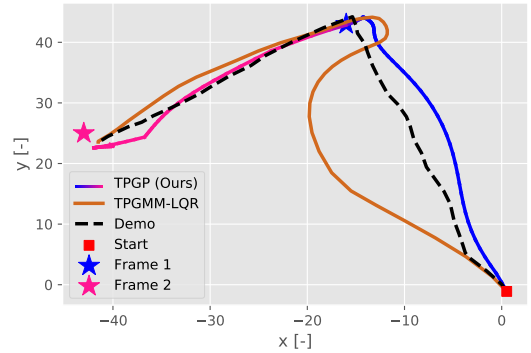
For the three frame task, the results are shown in Table III. Again, TPGP shows better performance according to almost all metrics. In the goal distance metrics, TPGP values are again less than half those of TPGMM-LQR in most cases.

Example trajectories generated by both models are shown in Figures 12a and 12b for a training and test configuration, respectively. Both of these figures qualitatively show how the trajectories generated by the proposed model are more efficient and close to straight-line routes.

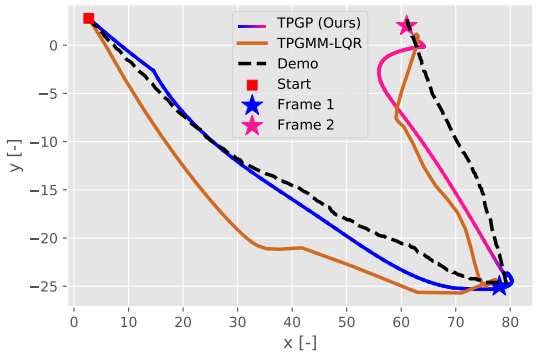
VI. Validation on a Robotic Manipulator

The experimental setup is first explained in Section VI-A. Section VI-B shows the results of TPGP on a re-shelving task, and Section VI-C compares the performance of TPGP and TPGMM-LQR using the demonstration data from this task.

[1]An implementation can be found at https://gitlab.idiap.ch/rli/pbdlib-python/-/blob/master/notebooks/pbdlib%20-%20Multiple%20coordinate%20systems.ipynb?ref_type=heads



(a) Generated trajectories for a training configuration.

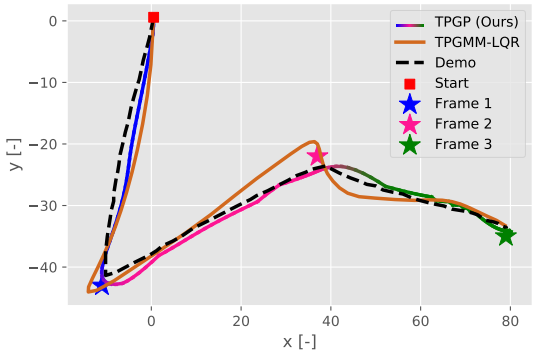


(b) Generated trajectories for a held-out test configuration.

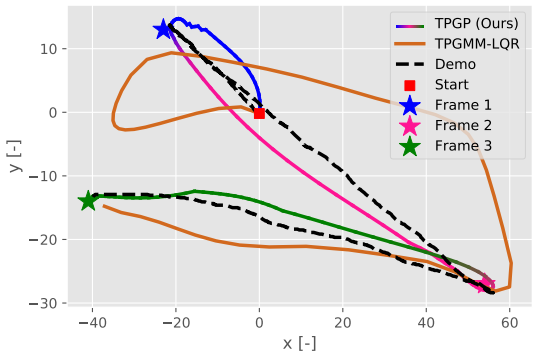
Fig. 11: Generated trajectories by TPGP (in blue and pink) and TPGMM-LQR (in orange) for the two frame task. The color of the TPGP trajectory visualizes the relevance weight output by the classifier, with blue corresponding to frame 1 and pink to frame 2. The demonstrations are shown as black dashed lines. Note the deviation of TPGMM-LQR from the demonstration for the training case.

TABLE III: Performance metrics for TPGP and TPGMM for the three frame task. For all metrics, lower values indicate better performance. Bold values indicate that the values used to calculate the average were significantly lower according to the Mann-Whitney U test.

	Average distance to goal 1 [-]		Average distance to goal 2 [-]		Average distance to goal 3 [-]		Average Fréchet distance [-]	
	train	test	train	test	train	test	train	test
TPGMM-LQR	2.69	4.32	1.07	1.45	2.83	2.52	12.6	13.9
TPGP (Ours)	**0.71**	**0.93**	**0.84**	**0.66**	**0.35**	**0.35**	**5.2**	14.43



(a) Generated trajectories for a training configuration.



(b) Generated trajectories for a held-out test configuration.

Fig. 12: Generated trajectories by TPGP (in blue, pink, and green) and TPGMM-LQR (in orange) for the three frame task. The color of the TPGP trajectory visualizes the relevance weight output by the classifier, with blue corresponding to frame 1, pink to frame 2, and green to frame 3. The demonstrations are shown as black dashed lines. Note how TPGMM-LQR deviates from the demonstration path for the test case.

A. *Experimental Setup*

TPGP is also tested and validated using a 7-degree-of-freedom robotic manipulator. Cartesian impedance control is used [23] to control the robot, where the end-effector is modeled as a spring-damper system. Thus to provide kinesthetic demonstrations the stiffness can be set to zero and the demonstrator is free to move the robot.

At execution time, TPGP is used in an offline fashion: a trajectory is first generated and then executed by the controller as a sequence of attractors. For the re-shelving task, it is also necessary to generate values for the orientation and gripper commands of the end-effector. A similar strategy to that of the progress variable increment $\Delta\varphi$ is used: at each timestep i, the most correlated point from the training data $\boldsymbol{X}$ to the current state $\boldsymbol{x}_i$ (thus including both position and progress value) is found. The recorded gripper value and orientation at that most correlated training data point are then used for that timestep.

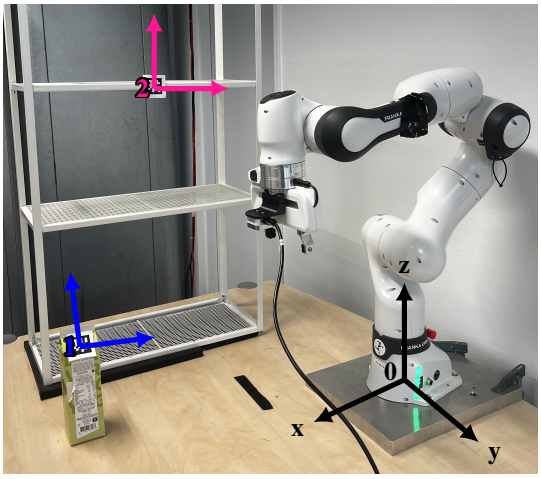


Fig. 13: The robotic setup with the relevant base, milk, and shelf frames indicated as frames 0, 1, and 2, in black, blue, and pink, respectively. The positions of the milk and shelf goal are perceived through the camera attached to the end-effector using the AprilTags visible in the image.

TABLE IV: Performance metrics for TPGP trained with an increasing number of demonstrations.

	Average distance to goal 1 [cm]		Average distance to goal 2 [cm]		Average Fréchet distance [cm]		Task success rate [-]
	train	test	train	test	train	test	
4 Demos	1.0	1.2	2.4	3.1	10.2	20.5	50%
6 Demos	0.9	3.7	1.3	1.2	5.6	17.3	70%
8 Demos	0.5	3.8	1.2	2.5	8.3	10.6	80%
10 Demos	1.0	3.5	1.2	2.5	7.0	10.0	100%

TABLE V: Performance metrics for TPGP without variance minimization trained on 4 and 10 demonstrations. For all metrics, lower values indicate better performance.

	Average distance to goal 1 [cm]		Average distance to goal 2 [cm]		Average Fréchet distance [cm]		Task success rate [-]
	train	test	train	test	train	test	
4 Demos w/o var. min.	1.2	6.8	3.0	6.1	8.6	22.4	10%
10 Demos w/o var. min.	1.0	4.0	1.3	1.3	11.0	20.0	80%

B. Pick and place task experiments

Using a robotic manipulator, the TPGP framework is now tested on a pick-and-place re-shelving task. A carton of milk must be picked up and then placed on a specific location on a shelf. Fiducial markers (AprilTags [24]) are used to localize the carton of milk and the placing goal during demonstrations and at execution time. An image of the setup for this task is shown in Figure 13, where the base frame as well as the milk (frame 1) and placing goal (frame 2) are indicated on top of the fiducial markers.

First, 10 demonstrations of this task are recorded while varying the position of both the carton and the placing goal. Another 5 demonstrations are recorded to be used as a test set. Several TPGP models are then trained using four, six, and eight randomly chosen demonstrations out of the training demonstrations, and their performance is compared to a model trained with the complete dataset.

The results are shown in Table IV, where the same performance metrics explained in Section V are used. Additionally, each of the models is used to try and complete the task for 10 configurations of both frames, and the success rate is reported in the last column. Note how the metrics mostly improve as the number of demonstrations increases, especially the success rate, which could be considered the most important metric in this case.

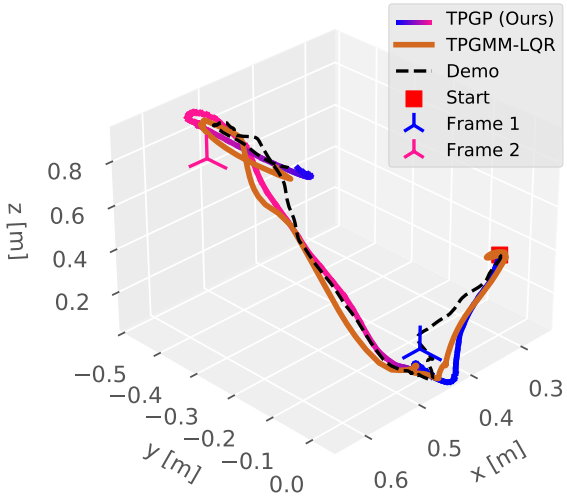
Similarly to the experiments in Section V-B, the performance metrics are again reported for this task for a model with no variance minimization in the local policies. The results are shown in Table V for the model trained on 4 demonstrations and for the model trained on all 10. These models without variance minimization are also used to try and complete the task for 10 new frame configurations, and the task success rate is reported. Note how, again, the values for these models are worse compared to the models with variance minimization (see Table IV). Moreover, these results also show that variance minimization is especially helpful when fewer demonstrations are available.

C. Comparison with TPGMM

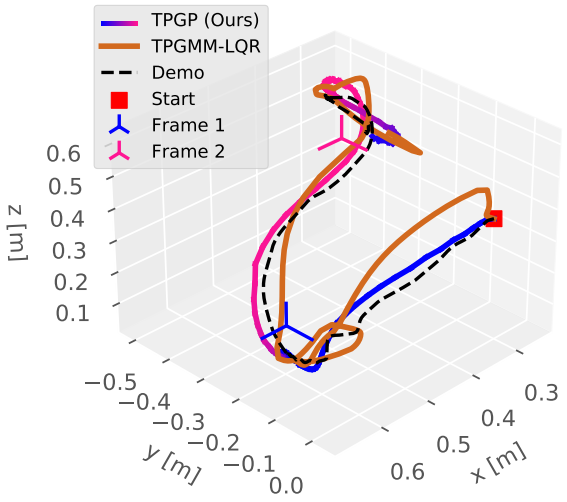
A comparison with TPGMM-LQR is presented in this subsection, but now using the demonstration data recorded with the robotic manipulator for the re-shelving task. Note that for the results of the previous subsection, it was possible to fully perform the task using the TPGP model by using the correlation functions from the learned local GPs plus the classifier weights to generate orientation and gripper commands. Since this is not possible for the TPGMM-LQR model, the comparisons presented here do not include success rates. Instead, the same performance metrics explained in Section V are used to compare the models' output trajectories for the training configurations and the held-out test demonstrations. Some qualitative comparisons of the trajectories are also presented.

Examples of generated trajectories for training and test configurations are shown in Figures 14a and 14b, respectively. The comparison results are shown in Table VI. Note how no values are bold for the goal distance metrics, which means there was no statistically significant difference between the values according to the U test, and thus both models could be said to have similar performance for these metrics. Nonetheless, TPGP does perform better in Fréchet distance for both the training and test sets.

Despite both models having similar performance according to these performance metrics, it is also interesting to evaluate the generated trajectories qualitatively. For example, while the TPGMM-LQR-generated trajectories meet the placing goal up to an error similar to that of TPGP, some of these trajectories



(a) Generated trajectories for a training configuration.



(b) Generated trajectories for a held-out test configuration.

Fig. 14: Generated trajectories by TPGP (in blue and pink) and TPGMM (in orange) for the re-shelving task with the robot. The color of the TPGP trajectory visualizes the relevance weight output by the classifier, with blue corresponding to frame 1 and pink to frame 2. The demonstrations are shown as black dashed lines. Note the deviation of TPGMM-LQR from the demonstration in the test case when close frame 1.

TABLE VI: Performance metrics for TPGP and TPGMM for pick and place task. For all metrics, lower values indicate better performance. Bold values indicate that the values used to calculate the average were significantly lower according to the Mann-Whitney U test.

	Average distance to goal 1 [cm]		Average distance to goal 2 [cm]		Average Fréchet distance [cm]	
	train	test	train	test	train	test
TPGMM-LQR	0.6	4.4	0.6	1.0	13.2	12.4
TPGP (Ours)	0.8	4.7	1.0	1.3	**8.9**	**9.7**

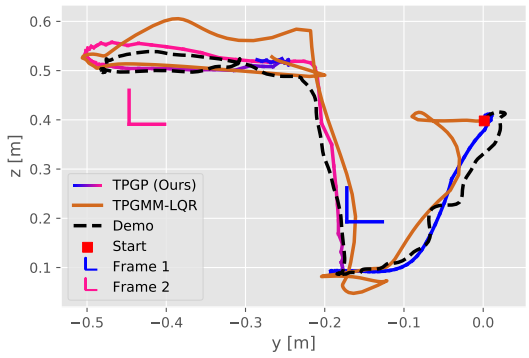


Fig. 15: Side view of generated trajectories by TPGP and TPGMM for re-shelving task on the robot. The color of the TPGP trajectory visualizes the relevance weight output by the classifier, with blue corresponding to frame 1 and pink to frame 2, and the corresponding demonstrations are shown as black dashed lines. Note how TPGMM-LQR deviates from the demonstration in height (z-coordinate) close to the placing goal above frame 2.

deviate considerably from the demonstrated trajectories at other points. Viewing an example trajectory from the side, shown in Figure 15, a deviation in height (z-coordinate) of at least 5 cm can be observed right before the placing goal, close to frame 2. In some cases, such a deviation could be enough for the milk carton to collide with the upper shelves, potentially causing a failure of the task.

Similarly, a deviation of more than 10 cm again occurs close to frame 1 for the TPGMM-LQR trajectory shown in 14b, close to the point in the trajectory where the carton would be picked. In a more realistic scenario with other cartons next to the target carton to be picked, this deviation could result in an undesired collision.

VII. Conclusions and Future Work

A framework is presented to learn multi-reference frame skills directly from demonstration, without using task-specific heuristics to pre-segment the demonstrations or to select reference frames at execution time by using local policies and

a self-supervised approach to train a classifier that determines frame relevance to select from these local policies. It is shown that the framework can learn simple point-to-point movements, as well as more complex skills. It is also able to learn skills involving more than two frames and generalize to unseen configurations. The framework is also validated on a robotic manipulator for a re-shelving task, where the model successfully performs the task for new, unseen configurations.

For both, simulation data and robot data, the framework is compared with the performance of another segmentation- and heuristic-free model, TPGMM-LQR. In both cases, the TPGP model shows better performance in the majority of the given metrics. Ablation experiments show the importance and utility of variance minimization in the local policies, as well as the additional variance term J_g when training the classifier. Both of these features importantly make use of the uncertainty quantification of Gaussian Processes (GPs).

To extend the generalization capabilities of the framework for robotic manipulation tasks, it would also be interesting to train local policies on the robot's orientation data. In some cases, including the orientation data could also improve the classifier's performance when the position information is ambiguous.

Another interesting avenue for future work could be to explore alternative inputs for both the local policies and the classifier. As explained in Section IV-C, using only the current position leads to ambiguities, which is why the progress value is necessary. However, instead of adding an explicit progress value, some form of memory could be added to the model, for example by including previous states or a hidden state that encodes this information. Similarly, it was shown in Section IV-D that adding the current position relative to each of the frames to the input of the classifier also leads to ambiguities. Again, adding some form of memory could address these ambiguities.

Making this modification to both the local policies and the classifier would enable the model to learn a wider variety of skills that are not necessarily dependent only on timing. Moreover, it could address some limitations in performance for the tasks shown in this work. For example, when the distances between frames are too different compared to those seen during the training, the model might switch too early or too late. Making the model less dependent on timing could also make it more resistant to disturbances. An alternative encoding that is less time-dependent could also enable the re-sequencing of learned local policies to perform new skills without having to provide new demonstrations.

Finally, a limitation inherent to the proposed TPGP is the need for diverse demonstrations. If several demonstrations are given but the configurations of the frames are very similar among these demonstrations, the model will likely fail to generalize to new configurations. A related issue is that a demonstrator might not know how to ensure the demonstrations and configurations are diverse. Thus a possible extension would be to integrate TPGP in an incremental learning framework, where additional training can easily be provided. An active learning element could also be integrated where the algorithm can request additional training data, potentially making use again of the uncertainty quantification capability of Gaussian Processes.

REFERENCES

[1] M. Wächter and T. Asfour, "Hierarchical segmentation of manipulation actions based on object relations and motion characteristics," in *2015 International Conference on Advanced Robotics (ICAR)*, Jul. 2015, pp. 549–556. DOI: 10.1109/ICAR.2015.7251510.

[2] M. Mühlig, M. Gienger, and J. J. Steil, "Human-robot interaction for learning and adaptation of object movements," in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2010, pp. 4901–4907. DOI: 10.1109/IROS.2010.5649229.

[3] B. Li, J. Li, T. Lu, Y. Cai, and S. Wang, "Hierarchical Learning from Demonstrations for Long-Horizon Tasks," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 2021, pp. 4545–4551. DOI: 10.1109/ICRA48506.2021.9561408.

[4] J. Kober, M. Gienger, and J. Steil, "Learning movement primitives for force interaction tasks," *Proceedings - IEEE International Conference on Robotics and Automation*, vol. 2015, pp. 3192–3199, Jun. 2015. DOI: 10.1109/ICRA.2015.7139639.

[5] J. R. Flanagan, M. C. Bowman, and R. S. Johansson, "Control strategies in object manipulation tasks," *Current opinion in neurobiology*, vol. 16, no. 6, pp. 650–659, 2006.

[6] S. Manschitz, M. Gienger, J. Kober, and J. Peters, "Learning sequential force interaction skills," English, *Soft Robotics*, vol. 9, no. 2, 2020, ISSN: 2169-5172. DOI: 10.3390/ROBOTICS9020045.

[7] L. Pais, K. Umezawa, Y. Nakamura, and A. Billard, "Learning robot skills through motion segmentation and constraints extraction," in *HRI Workshop on Collaborative Manipulation*, Citeseer, 2013, p. 5.

[8] A. L. P. Ureche, K. Umezawa, Y. Nakamura, and A. Billard, "Task Parameterization Using Continuous Constraints Extracted From Human Demonstrations," *IEEE Transactions on Robotics*, vol. 31, no. 6, pp. 1458–1471, Dec. 2015, Conference Name: IEEE Transactions on Robotics, ISSN: 1941-0468. DOI: 10.1109/TRO.2015.2495003.

[9] S. Calinon, T. Alizadeh, and D. G. Caldwell, "On improving the extrapolation capability of task-parameterized movement models," in *Proc. IEEE/RSJ Intl Conf. on Intelligent Robots and Systems (IROS)*, Tokyo, Japan, Nov. 2013, pp. 610–616.

[10] S. Calinon, "A tutorial on task-parameterized movement learning and retrieval," *Intelligent service robotics*, vol. 9, pp. 1–29, 2016.

[11] M. Vlachos, G. Kollios, and D. Gunopulos, "Discovering similar multidimensional trajectories," Feb. 2002, pp. 673–684, ISBN: 0-7695-1531-2. DOI: 10.1109/ICDE.2002.994784.

[12] A. P. Ruiz, M. Flynn, J. Large, M. Middlehurst, and A. Bagnall, "The great multivariate time series classification bake off: a review and experimental evaluation of recent algorithmic advances," *Data Mining and Knowledge Discovery*, vol. 35, no. 2, pp. 401–449, 2021.

[13] M. Müller, "Dynamic time warping," *Information retrieval for music and motion*, pp. 69–84, 2007.

[14] M. Mühlig, M. Gienger, J. Steil, and C. Goerick, "Automatic Selection of Task Spaces for Imitation Learning," Oct. 2009, pp. 4996–5002. DOI: 10.1109/IROS.2009.5353894.

[15] C. Williams and C. Rasmussen, "Gaussian processes for machine learning, vol 2 Cambridge," *MA: MIT Press.*, 2006.

[16] J. Hensman, A. Matthews, and Z. Ghahramani, *Scalable Variational Gaussian Process Classification*, 2014. arXiv: 1411.2005 [stat.ML].

[17] G. Franzese, A. Mészáros, L. Peternel, and J. Kober, "ILoSA: Interactive Learning of Stiffness and Attractors," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2021, pp. 7778–7785. DOI: 10.1109/IROS51168.2021.9636710.

[18] A. Mészáros, G. Franzese, and J. Kober, "Learning to Pick at Non-Zero-Velocity From Interactive Demonstrations," *IEEE Robotics and Automation Letters*, vol. 7, no. 3, pp. 6052–6059, 2022. DOI: 10.1109/LRA.2022.3165531.

[19] G. Franzese, L. d. S. Rosa, T. Verburg, L. Peternel, and J. Kober, "Interactive Imitation Learning of Bimanual Movement Primitives," *IEEE/ASME Transactions on Mechatronics*, pp. 1–13, 2023. DOI: 10.1109/TMECH.2023.3295249.

[20] T. Eiter and H. Mannila, "Computing discrete Fréchet distance," 1994.

[21] H. B. Mann and D. R. Whitney, "On a test of whether one of two random variables is stochastically larger than the other," *The annals of mathematical statistics*, pp. 50–60, 1947.

[22] S. Calinon, "Robot Learning with Task-Parameterized Generative Models," in *Robotics Research: Volume 2*, A. Bicchi and W. Burgard, Eds. Cham: Springer International Publishing, 2018, pp. 111–126, ISBN: 978-3-319-60916-4. DOI: 10.1007/978-3-319-60916-4_7. [Online]. Available: https://doi.org/10.1007/978-3-319-60916-4_7.

[23] N. Hogan, "Impedance control: An approach to manipulation," in *1984 American control conference*, IEEE, 1984, pp. 304–313.

[24] J. Wang and E. Olson, "AprilTag 2: Efficient and robust fiducial detection," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct. 2016.

APPENDIX A
ALIGNMENT ALGORITHM

This appendix provides additional figures to show why Dynamic Time Warping (DTW) does not work in this case, and provides the LCSS algorithm as well as the alignment algorithm that uses LCSS to produce alignment points.

A. Dynamic Time Warping alignment

Figure 16 shows the alignment produced by DTW for two example demonstrations when they are in the base frame. Note how the annotated goals in each demonstration are matched with points very far from the goal in the other demonstration.

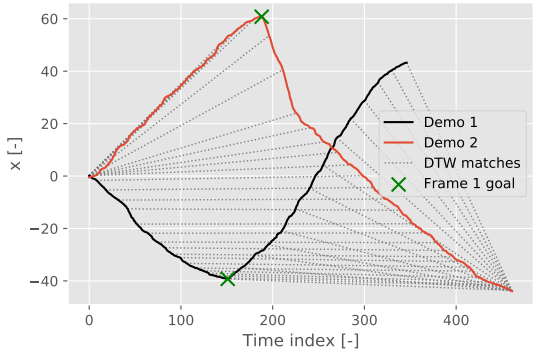


Fig. 16: Matching from DTW on base, non-transformed trajectories (only the x-coordinate is shown). Note how the marked frame 1 goals of each demo are not matched.

If the demonstrations are transformed to frame 1, the matching from DTW improves, as shown in Figure 17: note how the annotated goals are now matched with points close to the goal in the other demonstration. However, the rest of the matches would not necessarily produce a good alignment: the resulting alignment is shown in Figure 18, where demonstration 1 is warped according to the found DTW path. Note how the shape of the demonstration is now very different, even if the goals are closely aligned.

While this could be partially solved by imposing additional parameters on the DTW algorithm, for example, constraints on the alignment path, or removing the constraints to match all the initial and final points, this might not be very practical, since different parameters would probably be needed for different pairs of demonstrations. Moreover, the different alignments between different pairs of demonstrations would still need to be merged somehow, or one demonstration would have to be picked as a template to match every other demonstration with.

B. Longest Common Subsequence (LCSS) and alignment algorithms

The steps for the LCSS algorithm are shown in Algorithm 1. The alignment algorithm, shown in Algorithm 2, uses the LCSS algorithm as a function to find the LCSS path between pairs of demonstrations. Moreover, it uses a function

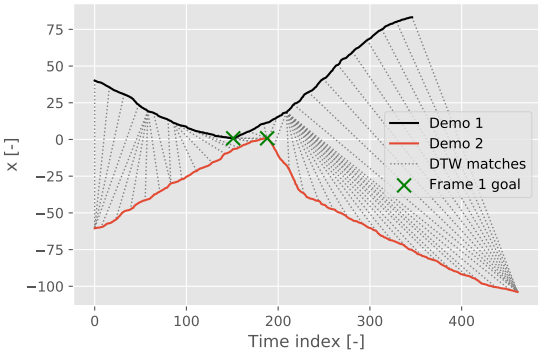


Fig. 17: Matching from DTW on trajectories transformed to frame 1 (only the x-coordinate is shown).

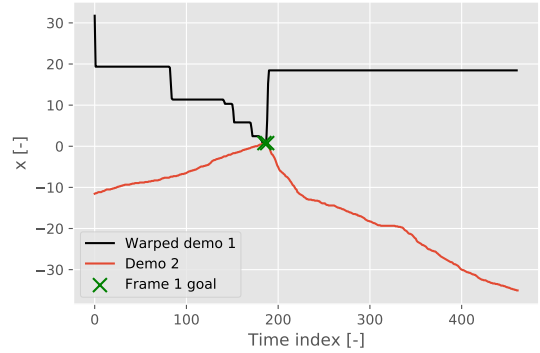


Fig. 18: Trajectory from demo 1 warped using DTW path and the unwarped demo 2 trajectory, both transformed to frame 1. Note how the warped demo 1 has lost a lot of its shape.

$\boldsymbol{MaxRepIdx}$ that takes as input separate lists of 2-tuples (essentially several LCSS paths) and returns the integer that occurs the most times among the input lists in the first position of the 2-tuples. The alignment algorithm iterates through each transformed dataset of demonstrations $\mathcal{D}^{(m)}$, and then also through all pairs of demonstrations in each dataset to find their LCSS paths, and ultimately returns the alignment points.

Algorithm 1 Longest Common Subsequence (LCSS)

Input: Sequences X, Y, similarity function **Sim**$(\cdot,\cdot)$, similarity threshold k
function **LCSS**(X, Y)
    $m \leftarrow$ length of X
    $n \leftarrow$ length of Y
    Initialize a 2D array L with dimensions $(m+1) \times (n+1)$
    for $i \leftarrow 0$ to m **do**
        for $j \leftarrow 0$ to n **do**
            if $i = 0$ or $j = 0$ **then**
                $L[i][j] \leftarrow 0$
            else if $\boldsymbol{Sim}(X[i-1], Y[j-1]) > k$ **then**
                $L[i][j] \leftarrow L[i-1][j-1] + 1$
            else
                $L[i][j] \leftarrow \max(L[i-1][j], L[i][j-1])$
            end if
        end for
    end for
    $lcssLength \leftarrow L[m][n]$
    $lcss \leftarrow$ empty list
    $i \leftarrow m$, $j \leftarrow n$
    while $i > 0$ and $j > 0$ **do**
        if $\boldsymbol{Sim}(X[i-1], Y[j-1]) > k$ **then**
            Add $X[i-1]$ to the beginning of $lcss$
            $i \leftarrow i - 1$
            $j \leftarrow j - 1$
        else if $L[i-1][j] > L[i][j-1]$ **then**
            $i \leftarrow i - 1$
        else
            $j \leftarrow j - 1$
        end if
    end while
    return $lcss, lcssLength$ ▷ LCS and its length
end function

Algorithm 2 Alignment points extraction algorithm

Input: Number of frames n_f, number of demonstrations n_d, and the sets of transformed demonstrations $\mathcal{D}^{(1)}, ..., \mathcal{D}^{(n_f)}$
Let $lcsspaths$ be an array with dimensions n_f x n_d x n_d, where each entry will store a list of integer 2-tuples
Let $alignpts$ be an array with dimensions n_d x n_f, where each entry stores an integer
for $k \leftarrow 0, n_f$ **do**
    for $i \leftarrow 0, n_d$ **do**
        for $j \leftarrow 0, n_d$ **do**
            $lcsspaths[k, i, j] \leftarrow \boldsymbol{LCSS}(d_i^{(k+1)}, d_j^{(k+1)})$
        end for
    end for
    for $i \leftarrow 0, n_d$ **do**
        $alignpts[i, k] \leftarrow \boldsymbol{MaxRepIdx}(lcsspaths[k, i])$
    end for
end for
return $alignpts$