Learning from Few Demonstrations with Frame-Weighted Motion Generation

Jianyong Sun

Master Thesis

TU Delft Delft University of Technology

Cognitive Robotics

Learning from Few Demonstrations with Frame-Weighted Motion Generation

Master Thesis

For the degree of Master of Science in Robotics
at Delft University of Technology

Student name:	Jianyong Sun	
Student number:	5278031	
Project duration:	November 15, 2021 – December 22, 2022	
Thesis committee:	Jens Kober,	TU Delft, Chair & Supervisor
	Jihong Zhu,	University of York, Daily supervisor
	Michael Gienger,	Honda Research Institute EU, Supervisor
	Luka Peternel,	TU Delft, External member
	Arno Stienen,	TU Delft, External member

Faculty of Mechanical, Maritime and Materials Engineering (3mE) · Delft University of Technology

Learning from Few Demonstrations with Frame-Weighted Motion Generation

Jianyong Sun

***Abstract*— Learning from Demonstration (LfD) aims to learn versatile skills from human demonstrations. The field has been gaining popularity since it facilitates transferring knowledge to robots without requiring much expert knowledge. During task executions, the robot motion is usually influenced by constraints imposed by environments. In light of this, task-parameterized (TP) learning encodes relevant contextual information in reference frames, enabling better skill generalization to new situations. However, most TP learning algorithms require multiple demonstrations in various environment conditions to ensure sufficient statistics for a meaningful model. It is not a trivial task for robot users to create different situations and perform demonstrations under all of them. Therefore, this paper presents a novel concept to learn motion policy from few demonstrations through *explicitly* solving reference frame weights along the task trajectory. Experimental results in both simulation and real robotic environments validate our approach.**

I. Introduction

Compared to industrial robots that perform repetitive tasks in well-defined working situations, service robots are expected to be dexterous and intelligent to work in variable and unstructured environments, learn skills quickly, and easily adapt to unseen scenarios. However, traditional task-specific robot programming requires enough expertise to program robot motions, which leads to the lower popularity of robots in society and poor adaptability to various task executions.

Inspired by the process that humans learn skills by imitating others, LfD enables robots to acquire versatile skills by extracting important and consistent features from expert demonstrations. It endows users with an intuitive interface to transfer new skills to robots without the need for time-consuming robot programming, and inefficient solution exploration [1]. To improve the generalization ability of robot learners, [2] proposes the TP learning method by parameterizing the description of task situation into reference frames. Nevertheless, to obtain enough statistical data, most existing TP learning methods require collecting multiple diverse demonstrations under different environment conditions. This is often not feasible in practice due to time and space limits, especially for complex tasks. With few demonstrations, these methods cannot extract useful features from training data, hence showing poor generalization performance.

To address this challenge, this paper proposes an approach to learning from few demonstrations through solving the weights of reference frames [1]. The frame weight can be interpreted as a measurement of its importance or relevance to the task. Benefiting from the calculated frame weights, the robot can generalize learned skills in novel situations by directly transforming the demonstrated trajectories. Despite reduced number of training demonstrations, robots can still have enough generalization capabilities to perform the task well under various conditions, either within or out of the range of their observed experience. Besides, for some tasks requiring the motion variance to quantify uncertainties, such as human-robot interaction scenarios, our proposed method allows the augmentation of training datasets.

The remainder of the paper is organized as follows. We first review related works about improving generalization and data efficiency in LfD and calculating the influence of task frames in Sec. II. We then detailedly state the problem we would solve in Sec. III. In Sec. IV, we introduce our proposed algorithm, which is then validated with simulation in Sec. V and robotic experiments in Sec. VI. We finally conclude in Sec. VII.

II. Related works

Generalization in LfD The generalization ability is considered one of the central properties of robot learning algorithms. Traditional movement encoding methods, such as Dynamic Movement Primitive (DMP), Probabilistic Movement Primitive (ProMP), and Gaussian Mixture Model (GMM), can normally only be applied in invariant situations, hence displaying limited extrapolation ability [3]. To improve that, many LfD algorithms integrate these methods with task parameters which can fully describe task situations [4]. Since task parameters encode relevant contextual information about the task, these algorithms usually allow the automatic adaptation of learned skills to various situations, showing better extrapolation performance.

Most of these methods formulate the motion retrieval from task parameters as a regression problem. In [5], Gaussian Process Regression (GPR) is used to model the mapping from task parameters to DMP parameters which represent robot trajectories compactly. Similarly, [6] presents how to utilize the mixture of GMMs to model the joint distribution of task parameters and nonlinear forcing terms in DMP. [7] applies Locally Weighted Regression (LWR) to model the conditional distribution of the demonstrated trajectories with respect to the demonstration contexts.

A prominent TP learning framework using local reference frames as task parameters was proposed in [4] by exploiting the spatial transformation between frames to achieve better extrapolation performance, giving task-parameterized Gaussian Mixture Model (TP-GMM). The approach first projects

[1]We will use frame weight, influence, importance and relevance interchangeably.

the global trajectories into different local reference frames, then extracts locally consistent features among demonstrations using GMMs, and finally transfers them into new situations leveraging the linear transformation property of Gaussian distributions. While these methods with task parameters utilized perform better in generalization compared with traditional LfD algorithms, most still require a number of demonstrations which are not always easy to collect.

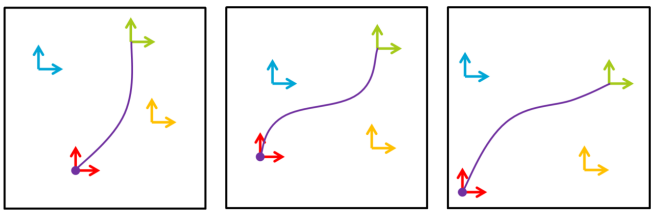
Date efficiency in LfD Since data efficiency is important to LfD algorithms, several methods have been proposed to decrease their dependence on the amount of data. [8] uses Corrected Augmentation for Trajectories (CAT) to correct distorted expert actions, which can augment original expert demonstrations. The routine-augmented policy learning (RAPL) framework proposed by [9] discovers routines from a single demonstration and uses them to augment policy learning. [10] augments the original demonstration dataset with the data synthesized through noise injection, TP-GMM generation, and both. As a noise injection variant of Behavior Cloning (BC), Disturbances for Augmenting Robot Trajectories (DART) provides a broader set of state-action pairs by adding noise during demonstration collection phases [11]. While these methods help reduce the number of demonstrations required, most of them focus on augmenting the original training dataset, which introduces a lot of extra effort. Instead, our proposed algorithm provides the possibility to learn directly from the existing few demonstrations.

Frame weight calculation In most robotic tasks, the robot movements are modulated by a set of local reference frames with varying weights which capture frame importance along the whole trajectory. Taking the grasping task as an example, as the robot gets closer to the target, its motion is more influenced by the reference frame attached to the target.

Most of the existing methods to solve frame weights build on TP-GMM. In [12], the frame importance is computed as the ratio of the precision matrix determinant with respect to the other frames. [13] and [14] utilize this definition to identify relevant frames to the task. Instead of solving the frame weight from data, [15] proposes using a confidence weight to directly modify the covariance matrix of each local model. But its performance is limited by the reliance on the human prior information. By combining the above two schemes, [16] calculates the confidence weight using the norm of data covariance modified with an extra shape parameter. However, these methods still have a strong dependence on the amount of data. In contrast, our method can calculate the frame weight using few demonstrations.

III. Problem Statement

Our goal is to find how the robot movements are modulated by reference frames. For example, Fig. 1 displays four demonstrations for a simple movement task. Obviously, the robot motion is only influenced by the red and green frames. The other two frames can be considered irrelevant to this task. Furthermore, the influence of two relevant frames changes along the trajectory. While the starting motion is more influenced by the red frame, the green frame is more important to the ending part.



(a) (b) (c)

Fig. 1: A simple movement task with four reference frames provided. Three demonstrations are displayed with different frame positions. The demonstrated trajectory is shown in purple with the start denoted by a dot.

We consider P local reference frames and refer to the rotation matrix $\boldsymbol{A}_j \in SO(p)$ and translation vector $\boldsymbol{b}_j \in \mathbb{R}^p$ of each frame $\{j\}$ with respect to the global coordinate system as task parameters, where $j = 1, 2, \ldots, P$, and p is 2 or 3 depending on the dimensionality required for the robot task.

The training dataset $\{\{\boldsymbol{\xi}_{t,m}\}_{t=0}^{T_m}\}_{m=1}^{M}$ consists of M demonstrations, each with the time length T_m which may be different for each demonstration. Their associated reference frames are denoted as $\{\{\boldsymbol{A}_{j,m}, \boldsymbol{b}_{j,m}\}_{j=1}^{P}\}_{m=1}^{M}$. Here, for each demonstration m, $\boldsymbol{\xi}_{t,m} \in \mathbb{R}^p$ represents a p-dimensional trajectory point recorded at time t. Considering the frame weight captures its importance along the whole trajectory, we calculate the relative distance $d_{t,m}$ of each point $\boldsymbol{\xi}_{t,m}$ as follows:

$$d_{t,m} = \frac{\|\boldsymbol{\xi}_{t,m} - \boldsymbol{\xi}_{0,m}\|}{\|\boldsymbol{\xi}_{t,m} - \boldsymbol{\xi}_{0,m}\| + \|\boldsymbol{\xi}_{t,m} - \boldsymbol{\xi}_{T_m,m}\|}, \tag{1}$$

where $\|\cdot\|$ represents the Euclidean distance between the data point and the start or end of the trajectory.

To solve the frame influence on robot motion, for each reference frame $\{j\}$, we wish to find a function f_j which maps the relative distance along the trajectory to the frame weight W_j, i.e.

$$f_j : d \longmapsto W_j, \tag{2}$$

where W_j should be between 0 and 1 and the weights of all reference frames should sum to one for the same d. Since we do not have the frame influence W_j in hand, it seems impossible to solve this function through regression. So we need to find some additional constraints, which will be introduced in Sec. IV.

IV. Methods

We introduce the proposed approach to calculating frame weights in this section. We start with an intuitive description by giving a simple introductory example in Sec. IV-A, and then a more general explanation of the algorithm step by step from Sec. IV-B to Sec. IV-D.

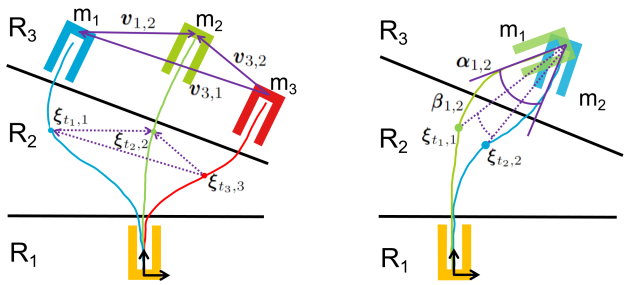


Fig. 2: A simulated 2D movement task from the start (the yellow U-shape box, frame $\{1\}$) to the target (the blue, green and red U-shape boxes, frame $\{2\}$). For simplicity, only the frame attached to the yellow box is shown. Separated by black lines, R_1, R_2, and R_3 denote the trajectory segments near the start (in the yellow box), in the middle (between two boxes), and near the goal (in the target box). m_i represents a demonstration. (a) Three demonstrations with targets in the same orientation but different positions. (b) Two demonstrations with targets in the same position but different orientations.

A. Intuitive Illustration

Rather than calculating the frame importance using variance information, we find how the reference frame influence changes along the trajectory in a more explicit way. To illustrate the idea better, Fig. 2 presents a 2D movement task from one U-shape box to the other without any collision with them. The shape of boxes imposes geometric constraints on the starting and ending parts of the motion. Displayed as dots, the corresponding points $\boldsymbol{\xi}_{t_1,m_1}$, $\boldsymbol{\xi}_{t_2,m_2}$, and $\boldsymbol{\xi}_{t_3,m_3}$ on different trajectories can be found by minimizing their relative distance error, such as $|d_{t_1,m_1} - d_{t_2,m_2}|$.

In Fig. 2a, the solid purple lines indicate the target position difference vectors $\boldsymbol{v}_{1,2}$, $\boldsymbol{v}_{3,1}$, $\boldsymbol{v}_{3,2}$ between demonstrations. The dashed purple lines represent the position difference vectors between corresponding points. In Fig. 2b, two solid purple lines form the target angle difference $\alpha_{1,2}$. Two dashed purple lines connecting a corresponding point to the frame origin form the angle difference $\beta_{1,2}$.

With the observation that the robot trajectory is modulated by the reference frames in mind, we find there exists a scaling factor s such that

$$\begin{cases} \boldsymbol{\xi}_{t_2,2} - \boldsymbol{\xi}_{t_1,1} = s\,\boldsymbol{v}_{1,2}, \\ \boldsymbol{\xi}_{t_2,2} - \boldsymbol{\xi}_{t_3,3} = s\,\boldsymbol{v}_{3,2}, \\ \boldsymbol{\xi}_{t_1,1} - \boldsymbol{\xi}_{t_3,3} = s\,\boldsymbol{v}_{3,1}, \\ \boldsymbol{\beta}_{1,2} = s\,\alpha_{1,2}. \end{cases} \tag{3}$$

In region R_1, the factor s is close to 0 because the robot motion is almost fully constrained by the starting box. Along the trajectory to the target, this factor gradually increases with the influence of frame $\{2\}$ getting stronger. Finally, in region R_3, it reaches 1 due to the full constraint of the ending box. This factor s exactly represents the frame $\{2\}$ influence.

Inspired by Eq. (3), given a reference trajectory, we can generalize it to other situations by transforming it using frame weights. Besides, we assume the trajectories in the same situation should be similar. Therefore, one expert demonstration should be as similar as possible to its reconstructed version from another one, which provides a implicit constraint on frame weights. Then we can solve them by minimizing the difference between ground truth and its reconstruction using a reference trajectory. Solving the frame influence function f_j in Eq. (2) is finally transformed into an optimization problem.

To simplify the calculation of frame weights for a multi-frame task ($P \geq 3$), we would divide the whole task into a sequence of subtasks with only two reference frames involved. The demonstration is still collected from start to finish, so the key issue is to segment it into component trajectories modulated by only two frames. For simple tasks, this can be achieved based on human prior information. For more complex tasks, this challenge has been well-studied and addressed [1]. For example, [17] proposes using a Beta-Process Auto-regressive HMM to achieve automatic recognition of repeated skills and segmentation of whole demonstrations. Therefore, in the following, we will focus on the two-frame task.

B. Demonstration Reconstruction

Given two demonstrations m_1 and m_2, we now introduce how to reconstruct m_1 from m_2 utilizing frame weights. We first project the global demonstrations into the local frame $\{1\}$ so that we only need frame $\{2\}$ weights for reconstruction since the difference between local trajectories $\boldsymbol{\xi}^{(1)}_{t,m}$ only result from different $\{\boldsymbol{A}^{(1)}_{2,m}, \boldsymbol{b}^{(1)}_{2,m}\}$.

1) Position Difference: Similar to Fig. 2a, we first consider the difference of frame positions $\boldsymbol{b}^{(1)}_{2,m_1}$ and $\boldsymbol{b}^{(1)}_{2,m_2}$ between two demonstrations. For each data point $\boldsymbol{\xi}^{(1)}_{t,m_2}$, we can transform it as follows:

$$\tilde{\boldsymbol{\xi}}^{(1)}_{t,m_1} = \boldsymbol{\xi}^{(1)}_{t,m_2} + f_2(d_{t,m_2})\,(\boldsymbol{b}^{(1)}_{2,m_1} - \boldsymbol{b}^{(1)}_{2,m_2}), \tag{4}$$

where $\tilde{\boldsymbol{\xi}}^{(1)}_{t,m_1}$ is one reconstruction point of the demonstration m_1 and $f_2(d_{t,m_2})$ represents the weight of frame $\{2\}$. Similar to Eq. (3), the required position transformation of two corresponding points on two trajectories should be the position difference of frame $\{2\}$ multiplied by a weight which incorporates its importance at this relative distance.

2) Orientation Difference: Subsequently, we need to consider the influence of frame orientation on the trajectory. Similar to the target angle difference $\alpha_{1,2}$ shown in Fig. 2b, we need to first calculate the difference of frame orientations $\boldsymbol{A}^{(1)}_{2,m_1}$ and $\boldsymbol{A}^{(1)}_{2,m_2}$ as follows

$$\boldsymbol{u}_{1,2} = g(\boldsymbol{A}^{(1)}_{2,m_1}\boldsymbol{A}^{-1}_{2,m_2}), \tag{5}$$

where $\boldsymbol{u}_{1,2}$ is the representation of the rotation vector and the function g can transform a rotation matrix into a vector. The vector direction indicates the rotation axis, and its magnitude

represents the angle. We then obtain the required rotation transformation

$$\boldsymbol{T}_{1,2} = h(f_2(d_{t,m_2})\, \boldsymbol{u}_{1,2}), \tag{6}$$

where h is a function to achieve the opposite conversion to the function g. Multiplying $\boldsymbol{u}_{1,2}$ by a frame importance scalar $f_2(d_{t,m_2})$, we can get the required rotation vector whose direction is the same as $\boldsymbol{u}_{1,2}$ but whose magnitude changes depending on the frame weight.

Finally, to make the reconstructed trajectory meet the orientation constraints of frame $\{2\}$ of m_1, we need to rotate the reconstructed point $\tilde{\boldsymbol{\xi}}^{(1)}_{t,m_1}$ around the origin of frame $\{2\}$ as follows

$$\hat{\boldsymbol{\xi}}^{(1)}_{t,m_1} = \boldsymbol{T}_{1,2}(\tilde{\boldsymbol{\xi}}^{(1)}_{t,m_1} - \boldsymbol{b}^{(1)}_{2,m_1}) + \boldsymbol{b}^{(1)}_{2,m_1}, \tag{7}$$

where $\hat{\boldsymbol{\xi}}^{(1)}_{t,m_1}$ is the final reconstruction point based on the frame pose difference.

After transforming each data point $\boldsymbol{\xi}^{(1)}_{t,m_2}$ in the demonstration m_2 following Eq. (4) and Eq. (7) sequentially, and transferring them back to the global coordinate frame, we finally get the reconstructed demonstration $\hat{m}_1 = \{\hat{\boldsymbol{\xi}}_{t,m_1}\}_{t=0}^{T_{m_2}}$.

C. Frame Weight Calculation

In this subsection, we illustrate how to solve frame weights based on demonstration reconstructions. To reduce the number of variables to be solved and simplify the calculation of frame weight, we approximate the function f_j in Eq. (2) as a weighted summation of Radial Basis functions (RBFs)

$$f_j(d) = \sum_{i=1}^{Q} \omega_{j,i}\, exp(-h_i(d - c_i)^2), \tag{8}$$

where Q is the number of required RBFs, $\omega_{j,i}$ represents the component weight of each basis function, and h_i and c_i denote the spread and center of the i-th basis function, respectively. In this case, after specifying the parameters h_i and c_i, we transform the problem of finding a function as the calculation of Q basis function weights.

After obtaining the reconstructed demonstration as presented in Sec. IV-B, we can solve frame weights by minimizing its difference to the ground truth. Here, we define the dissimilarity between two demonstrations as their normalized distance computed by dynamic time warping (DTW) (see [18] for details)

$$D(\hat{m},\, m) = \min_{\phi} d_{\phi}(\hat{m},\, m), \tag{9}$$

where ϕ is the warping function which maps the data point in $\hat{m}$ to m with minimum dissimilarity d.

With M demonstrations in the training dataset, we can find at most $N = \frac{M(M-1)}{2}$ pairs $\{k_1, k_2\}_{k=1}^{N}$ for trajectory reconstruction. k_1 and k_2 represent two demonstrations in the k-th pair. For each pair, we reconstruct k_1 from k_2. Then, we can formulate the calculation of frame influence as an optimization problem

$$\min_{\{\omega_{i,2}\}_{i=1}^{Q}} \frac{1}{N} \sum_{k=1}^{N} D(\hat{k}_1,\ k_1) \tag{10}$$

$$s.t. \quad 0 \le f_2(d) \le 1, \tag{11}$$

where we would find the optimized basis function weights determing the frame influence in Eq. (8) through minimizing the dissimilarity between one ground truth demonstration and its reconstructed version from another reference demonstration using the frame influence. We also add the constraint so that the solved frame weight should be between 0 and 1. Several popular optimization algorithms can be used to solve this problem, such as Sequential Least Squares Programming (SLSQP).

We previously assume the weights of all reference frames influencing the task trajectory should sum to one at the same relative distance. Hence, after calculating the weight of frame $\{2\}$ through optimization, we can easily derive the weight of frame $\{1\}$ by subtracting the calculated frame $\{2\}$ weights from one.

D. Skill Generalization

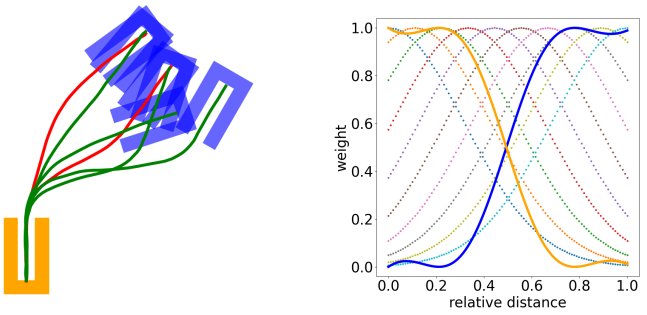
Once we have solved for the influences of each reference frame through optimization, we can directly transform a reference trajectory to generalize to new reference frames following the similar procedures described in Eq. (4) and Eq. (7). This process can be referred to as frame-weighted motion generation.

As a deterministic method, the frame-weighted motion generation can only provide an estimated trajectory without any variability. For some robotic tasks requiring variance information, we also need to evaluate the path uncertainty to determine how strongly the robot should follow the path. For example, in the human-robot interaction scenarios, we can dynamically adjust the robot stiffness based on the estimated variability for the sake of safe interaction. In this case, the frame-weighted motion generation can be used to augment the training dataset, which is also a promising application to help learn skills from few demonstrations. The augmented dataset can be encoded by probabilistic LfD algorithms, e.g., TP-GMM.

V. Simulation

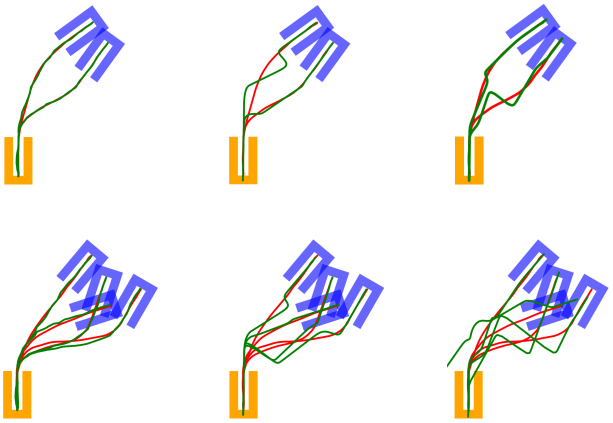
In this section, the proposed approach is evaluated on the simulated movement task shown in Fig. 2. We collect six expert demonstrations in total with different target positions and orientations, as shown in Fig. 3a. Two of them are used for training. To evaluate the extrapolation capability of models, we take the reaming four as the validation dataset. As introduced in Sec. IV-C, we still use the DTW normalized distance as the similarity measure between two trajectories. Then the model error can be defined as:

$$\mathcal{J}_{DTW} = \frac{1}{\mu} \sum_{i=1}^{\mu} \|\boldsymbol{y}_i - \boldsymbol{\xi}_i\|, \tag{12}$$



(a) (b)

Fig. 3: (a) The training (red) and validation (green) set for the simulated 2D movement task. Two reference frames are attached to the yellow (frame $\{1\}$) and blue (frame $\{2\}$) box, respectively. (b) The calculated weights of frame $\{1\}$ (yellow) and frame $\{2\}$ (blue) with RBFs (colorful dashed lines).



(a) (b) (c)

Fig. 4: The comparison of three motion generalization methods. The first and second row show generalizations in training and validation set. The generalizations and ground truths are marked with green and red. Column (a) Frame-weighted motion generation, (b) Augmented TP-GMM, (c) TP-GMM.

where $\boldsymbol{y}_i$ is the reproduced trajectory by models, $\boldsymbol{\xi}_i$ is the expert demonstration, and μ is the number of trials for generalization (2 for training and 4 for validation).

We now have three methods available for skill generalization:

- TP-GMM: the baseline method.
- Frame-weighted motion generation: to directly transform the reference trajectory in the training dataset utilizing frame weights.
- Augmented TP-GMM: to train TP-GMM on the training dataset augmented using frame-weighted motion generation.

For this simulation task, two reference frames are attached to the yellow and blue box shown in Fig. 3a, respectively. We train TP-GMM on time-based trajectories, so the reference frames need to be accordingly augmented with a time dimension. As a data preprocessing step, the trajectories used for TP-GMM training need to be temporally aligned using DTW [19]. The number of Gaussian components of TP-GMM is set to be 4. For augmented TP-GMM, we augment its training dataset to 8 demonstrations.

TABLE I: Training and validation error for three motion generalization methods

	Frame-weighted motion generation	Augmented TP-GMM	TP-GMM
Training error	0.011128 (32%)	0.040931 (116%)	0.035170 (100%)
Validation error	0.058802 (23%)	0.097286 (39%)	0.251572 (100%)

To approximate the frame weight function, we use 10 RBFs whose centers are uniformly distributed between 0 and 1 and whose spreads are all set to be 5. The optimized frame weights are depicted in Fig. 3b. The result is consistent with our intuitive understanding. At the starting part, the trajectory is fully constrained by the yellow box, so the frame $\{1\}$ weight reaches 1, and frame $\{2\}$ weight is almost 0. As the relative distance increases from 0 to 1, the influence of frame $\{2\}$ becomes larger due to the much closer distance to the blue box. Finally, when the trajectory reaches the blue box, the frame $\{2\}$ weight comes to 1 because of the geometric constraint of the box on the trajectory.

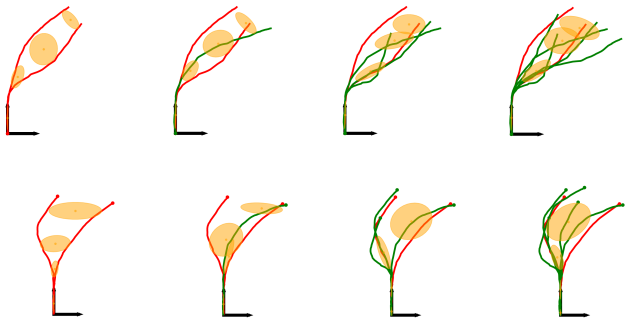
After using three motion generalization methods, we compute the error defined in Eq. (12) for the training and validation datasets. While the former indicates the model performance in extracting useful features from demonstrations and performing reproduction under observed conditions, the latter represents their ability to generalize to unseen situations. The results are summarized in Table I. We use the error of TP-GMM as the reference value and provide a percentage for each error value. The larger error is marked with red, and the smaller one with green. The corresponding generalized trajectories are displayed in Fig. 4. Since there are two demonstrations in the training dataset, we use the same reference trajectory for both frame-weighted motion generation and data augmentation [2].

From Table I, we observe that frame-weighted motion generation achieves the best performance, both for training and validation. Due to few training demonstrations, TP-GMM cannot obtain enough statistical data to effectively model the skill. In Fig. 4c, we can find the reproduced training demonstrations by TP-GMM are acceptable since the situations where GMR is used have been observed during training. However, the extrapolation performance of TP-GMM in the validation set is very poor because of the over-fitting to the few and sparse training data. In contrast, due to the explicit calculation and use of frame weights, our proposed algorithm can fully understand how the reference frames modulate the trajectory, enabling larger error reduction. Compared with TP-GMM, Fig. 4a shows that the

[2]In theory, either can be used as a reference. Here, to maintain consistency, we will always use the same one.

TABLE II: Average validation error of TP-GMM with increasing number of augmented demonstrations

No. of initial + new demos	2+0	2+1	2+2	2+3	2+4	2+5
Average validation error	0.252 (100%)	0.177 (70%)	0.153 (61%)	0.127 (50%)	0.112 (44%)	0.110 (44%)



(a) (b) (c) (d)

Fig. 5: The GMM of each frame changes with new demonstrations (green) added. The first and second rows show GMM in frame $\{1\}$ and $\{2\}$. The dots indicate the start of trajectories. The yellow ellipses represent the co-variance of each multivariate Gaussian. Column (a) initial demonstrations (red), (b) one new demonstration added, (c) three new demonstrations added, (d) five new demonstrations added.

generalized trajectories are much smoother and fulfill the geometric constraints of U-shape boxes.

With the training dataset augmented by 6 demonstrations generated using frame weights, TP-GMM also shows better performance in terms of the generalization on the validation set. The increasing training error indicates the overfitting of TP-GMM has been improved with more training data. Similar results can also be observed from Fig. 4b. Given 8 training demonstrations, the validation error of TP-GMM is still a little larger than frame-weighted motion generation. There are two possible reasons. First, all the augmented trajectories are generated using the same reference demonstration and frame weight, so TP-GMM is likely to over-fit a certain feature embodied by the reference. Additionally, the trajectories generated by each component of GMM through regression cannot be connected very smoothly. Obviously, in Fig. 4b, the generalized trajectories are all clearly displayed in 4 parts due to the use of 4 Gaussian components.

To further demonstrate the validity of our proposed method in terms of data augmentation, we show how the average validation error changes as the number of augmented trajectories increases. Here, the average means that we try various frame poses to augment the same number of demonstrations and calculate the average error of these trials. The results are presented in Table II. We can observe that the error reduction is most significant when the first new demonstration is provided and gradually decreases with more augmented demonstrations. In Fig. 5, we also show how the data distribution in each reference frame changes with more and more added demonstrations.

VI. Robotic Experiments

This section describes two experiments conducted to demonstrate the ability of our proposed method to learn skills from few demonstrations by solving the influence of reference frames on task trajectories.

A. Experimental Setup

As the experimental platform, we use two 7 DoF Franka Emika Panda placed vertically on a table and oriented in the same direction, see Fig. 6a. Their mutual poses can be calculated by locating the same object from two robot base frames and finding the optimal rigid transformation between two point sets [20]. The impedance controllers are deployed on both manipulators.

We perform 2 experiments with the real robot setup: *i*) a roller-on-holder task, where the right robot picks the paper roller from the holder on the table and places it on the other holder mounted on the left robot, see Fig. 6b and Fig. 6c, *ii*) a flower-in-vase task, where the right robot grabs the flower in the vase that vertically stands on the table and moves it to the other vase grasped by the left robot, see Fig. 6d and Fig. 6e. For both experiments, the left robot provides the target pose while the right one executes the task. Since we assume the right robot can automatically adjust the gripper orientation to align with the target during task execution, the trajectories we learn focus on the end-effector 3D positions of the right robot.

For both tasks, the right robot needs to adapt to different holder or vase poses which can be fully described by two reference frames attached to starting and ending ones:

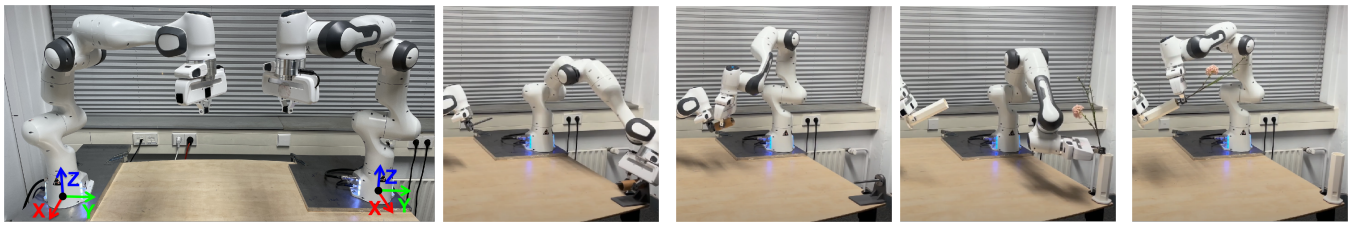
$$\{\boldsymbol{A}_s, \boldsymbol{b}_s\}, \quad \{\boldsymbol{A}_e, \boldsymbol{b}_e\}, \tag{13}$$

where $\boldsymbol{b}_s$ and $\boldsymbol{b}_e$ represent the bottom center positions of the holder or vase and the z axis of $\boldsymbol{A}_s$ and $\boldsymbol{A}_e$ specify their directions.

For both experiments, we first place the starting holder or vase on the table with a known pose, then record the end-effector poses of both robots during demonstrations. The latter is used for calculating the pose of the ending holder or vase and providing demonstrated trajectories. Since we utilize GMM to learn time-based trajectories, the reference frames used for TP-GMM training need to be accordingly augmented as follows:

$$\boldsymbol{A}' = \begin{bmatrix} 1 & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{A} \end{bmatrix}, \quad \boldsymbol{b}' = \begin{bmatrix} 0 \\ \boldsymbol{b} \end{bmatrix}. \tag{14}$$

Considering these two tasks are more complex than the simulated one, we increase the number of Gaussian components of TP-GMM to 6 and augment the training dataset to 9 demonstrations for augmented TP-GMM. For the weight calculation, we still use 10 RBFs, as introduced in Sec. V.

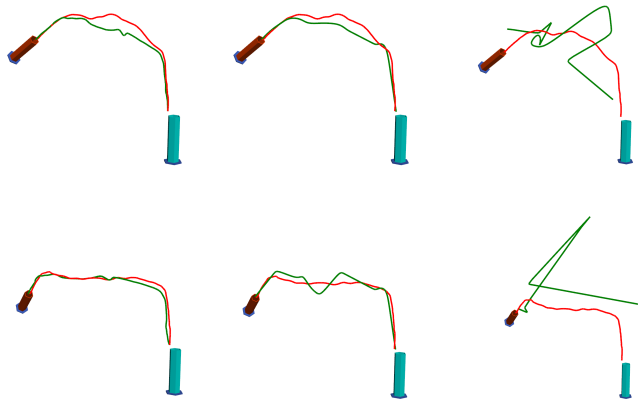


(a) (b) (c) (d) (e)

Fig. 6: The bimanual setup of two robot experiments. (a) shows two Franka Emika Panda robots on a table. The coordinate systems of robot bases are depicted with red, green, and blue (RGB) arrows. (b) and (c) display the initial and final state of the roller-on-holder task, while (d) and (e) present the initial and final state of the flower-in-vase task.

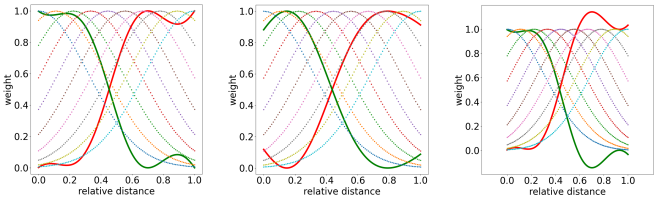
TABLE III: Training and validation error of three motion generalization methods for two real robotic tasks

	Roller-on-holder task			Flower-in-vase task		
	Frame-weighted motion generation	Augmented TP-GMM	TP-GMM	Frame-weighted motion generation	Augmented TP-GMM	TP-GMM
Training error	0.0151 (52%)	0.0385 (131%)	0.0293 (100%)	0.0209 (49%)	0.0511 (120%)	0.0427 (100%)
Validation error	0.0835 (13%)	0.1448 (22%)	0.6657 (100%)	0.0787 (11%)	0.1284 (18%)	0.7062 (100%)



(a) (b) (c)

Fig. 7: The validation performance of three generalization methods in the flower-in-vase task. Each row displays a validation situation. The starting and ending vases are shown in cyan and red. The generalized trajectories and ground truths are marked with green and red. Column (a) Frame-weighted motion generation, (b) Augmented TP-GMM, (c) TP-GMM.



(a) (b) (c)

Fig. 8: The solved frame weights of frame $\{1\}$ (green, starting vase) and frame $\{2\}$ (red, ending vase) through (un)constrained optimization. Column (a) with constraints, initialized with unconstrained optimization results, (b) with constraints, (c) without constraints.

B. Results

We test three skill generalization methods on both tasks. For each task, we use 2 and 4 expert demonstrations as the training and validation dataset. Table III presents the training and validation errors for both tasks.

Similar to the result shown in the simulated task, for both robotic experiments, the frame-weighted motion generation still achieves the best performance, even though the task is more complicated and the number of training demonstrations is still 2.

Taking the flower-in-vase task as an example, Fig. 7 presents how well these three methods perform in two validation situations. With only two training demonstrations, the initial TP-GMM cannot extract useful features from them, generating meaningless trajectories, as shown in Fig. 7c. When the training dataset is augmented by 7 new demonstrations, the performance is better. Although the two generated trajectories in Fig. 7b still display unwanted movements, they are successful in task executions since the geometric constraints of the vase have been satisfied. With Fig. 7a, we can intuitively observe how similar the generalized trajectory directly based on frame weights is to the ground truth.

In addition, we find the validation error reduction using frame-weighted motion generation in these two more complex tasks is larger than the simulated 2D task introduced in Sec. V. It means the frame weight incorporating how the reference frames modulate the task trajectories is one of the most important features for skill learning. For more complex tasks, this feature is more difficult to be extracted from training data, so TP-GMM with few demonstrations performs worse.

To further explore weight calculation, in Fig. 8, we present the frame weights solved through optimization with and without constraint that the weight value should be between

TABLE IV: Training and validation error using frame weights optimized with or without constraints

	Constrained optimization with initialization	Constrained optimization without initialization	Unconstrained optimization
Training error	0.014571 (116%)	0.015699 (125%)	0.012554 (100%)
Validation error	0.101014 (89%)	0.116217 (102%)	0.113575 (100%)

0 and 1. Without the constraint, the frame weight in Fig. 8c may be negative or exceed 1. This may be caused by data stochasticity since the weight is calculated using human demonstrations. But this phenomenon is contrary to the physical meaning of frame weights that the reference frame modulates task trajectories. Then, the solved frame weights using constrained optimization are displayed in Fig. 8b. Through Table IV, we can find the constrained optimization of weights causes larger training and validation errors. The former can be explained by the introduction of constraints. But the latter indicates that only the locally optimal solution is given. This problem can be addressed by initializing the constrained optimization with the result derived from unconstrained optimization. As shown in Fig. 8a, the frame weights seem more reasonable. In Table IV, the training and validation errors are both smaller than using constrained optimization without initialization. In particular, the validation error is also smaller than using unconstrained optimization, which indicates the assumption that the frame weight is between 0 and 1 is reasonable, and this human prior information helps improve the policy even further.

VII. Conclusion

Although task-parameterized learning in LfD achieves better generalization performance over non-parameterized alternatives, it requires multiple demonstrations collected in various situations which are not always available due to time and space constraints. This paper presents a method to learn skills from few demonstrations through the explicit calculation of reference frame weights. The method also allows augmentation of the original training dataset, enabling better performance of TP-GMM. We validate the method using a 2D simulation task and two real robot experiments.

We observe that the frame-weighted motion generation achieves much better performance compared to TP-GMM when the training demonstrations are few and sparse. Furthermore, the TP-GMM trained on the augmented training dataset still cannot reach the same performance as our proposed method. In addition, under the assumption that the frame weight incorporating its importance should be between 0 and 1, the constrained optimization of RBF weights provides more reasonable frame influence along the task trajectory and helps improve the extrapolation capability.

The current method is limited mainly by two aspects: *i*) due to little training data, the method requires demonstrations of high quality, *ii*) for multi-frame tasks, the segmentation of demonstrations into component skills depending on only two reference frames requires extra time and effort. The first problem can be easily solved by selecting high-quality demonstrations for training. For example, the demonstrations should be as smooth as possible, have less ambiguity, and follow the consistent human preference. The second limitation can be reduced with the help of state-of-the-art automatic demonstration segmentation algorithms [17] [21] [22].

References

[1] H. Ravichandar, A. S. Polydoros, S. Chernova, and A. Billard, "Recent advances in robot learning from demonstration," *Annual review of control, robotics, and autonomous systems*, vol. 3, pp. 297–330, 2020.

[2] S. Calinon, Z. Li, T. Alizadeh, N. G. Tsagarakis, and D. G. Caldwell, "Statistical dynamical systems for skills acquisition in humanoids," in *2012 12th IEEE-RAS International Conference on Humanoid Robots (Humanoids 2012)*. IEEE, 2012, pp. 323–329.

[3] S. Calinon and D. Lee, *Learning Control*, 01 2019, pp. 1261–1312.

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

[5] 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, pp. 1327–1339, 2012.

[6] A. Pervez and D. Lee, "Learning task-parameterized dynamic movement primitives using mixture of gmms," *Intelligent Service Robotics*, vol. 11, no. 1, pp. 61–78, 2018.

[7] T. Osa, A. M. G. Esfahani, R. Stolkin, R. Lioutikov, J. Peters, and G. Neumann, "Guiding trajectory optimization by demonstrated distributions," *IEEE Robotics and Automation Letters*, vol. 2, no. 2, pp. 819–826, 2017.

[8] D. Antotsiou, C. Ciliberto, and T.-K. Kim, "Adversarial imitation learning with trajectorial augmentation and correction," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 4724–4730.

[9] Z. Zhao, C. Gan, J. Wu, X. Guo, and J. B. Tenenbaum, "Augmenting policy learning with routines discovered from a single demonstration," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 12, 2021, pp. 11 024–11 032.

[10] J. Zhu, M. Gienger, and J. Kober, "Learning task-parameterized skills from few demonstrations," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 4063–4070, 2022.

[11] M. Laskey, J. Lee, R. Fox, A. Dragan, and K. Goldberg, "Dart: Noise injection for robust imitation learning," in *Conference on robot learning*. PMLR, 2017, pp. 143–156.

[12] T. Alizadeh, S. Calinon, and D. G. Caldwell, "Learning from demonstrations with partially observable task parameters," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2014, pp. 3309–3314.

[13] T. Alizadeh and M. Malekzadeh, "Identifying the relevant frames of reference in programming by demonstration using task-parameterized gaussian mixture regression," in *2016 IEEE/SICE International Symposium on System Integration (SII)*. IEEE, 2016, pp. 453–458.

[14] T. Alizadeh and B. Saduanov, "Robot programming by demonstration of multiple tasks within a common environment," in *2017 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI)*. IEEE, 2017, pp. 608–613.

[15] Y. Huang, J. Silvério, L. Rozo, and D. G. Caldwell, "Generalized task-parameterized skill learning," in *2018 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2018, pp. 5667–5474.

[16] A. Sena, B. Michael, and M. Howard, "Improving task-parameterised movement learning generalisation with frame-weighted trajectory generation," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2019, pp. 4281–4287.

[17] S. Niekum, S. Osentoski, G. Konidaris, and A. G. Barto, "Learning and generalization of complex tasks from unstructured demonstrations," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2012, pp. 5239–5246.

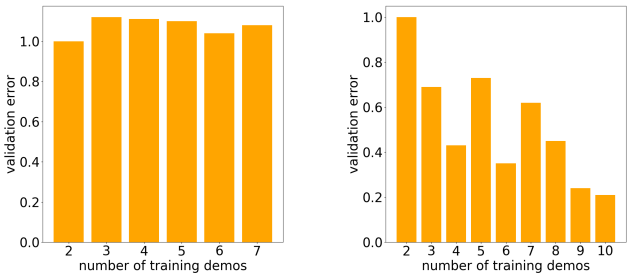
[18] M. Müller, "Dynamic time warping," *Information retrieval for music and motion*, pp. 69–84, 2007.

[19] S. Calinon, F. Guenter, and A. Billard, "On learning, representing, and generalizing a task in a humanoid robot," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 37, no. 2, pp. 286–298, 2007.

[20] K. S. Arun, T. S. Huang, and S. D. Blostein, "Least-squares fitting of two 3-d point sets," *IEEE Transactions on pattern analysis and machine intelligence*, no. 5, pp. 698–700, 1987.

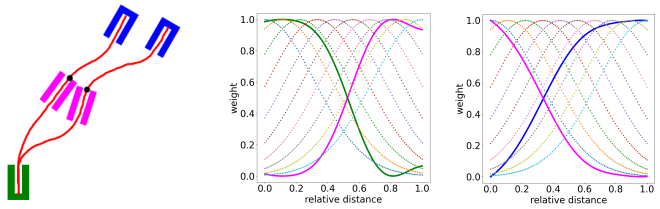
[21] O. Kroemer, H. Van Hoof, G. Neumann, and J. Peters, "Learning to predict phases of manipulation tasks as hidden states," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2014, pp. 4009–4014.

[22] R. Lioutikov, G. Neumann, G. Maeda, and J. Peters, "Probabilistic segmentation applied to an assembly task," in *2015 IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids)*. IEEE, 2015, pp. 533–540.



(a) (b)

Fig. 9: The validation error with more training demonstrations for the flower-in-vase task. (a) Frame-weighted motion generation. (b) TP-GMM.



(a) (b) (c)

Fig. 10: The simulated three-frame task. It can be described as moving from the green U-shape box (frame $\{1\}$), through the magenta pipe (frame $\{2\}$), and to the blue U-shape box (frame $\{3\}$). (a) displays two training demonstrations. The black dots indicates the segmentation of demonstrations. (b) displays the weights of frame $\{1\}$ (green) and $\{2\}$ (magenta) on the first sub-trajectory. (c) displays the weights of frame $\{2\}$ (magenta) and $\{3\}$ (blue) on the second sub-trajectory.

APPENDIX

A. Increasing training data

To compare our proposed method with TP-GMM in more depth, in Fig. 9, we display how the validation error changes with increasing training data. Fig. 9a indicates that the validation error using the frame-weighted generation remains almost unchanged. The reason behind this is the frame weights do not change as the number of demonstrations increases. In contrast, TP-GMM performs relatively better with more training demonstrations provided, as shown in Fig. 9b. This comparison indicates that our proposed method has less dependence on data, which further validates its ability to learn skills from few demonstrations.

B. Three-frame task: simulation

As shown in Fig. 10a, we set up a simulated three-frame task to evaluate the ability of our proposed method to deal with multi-frame tasks. We totally collect 8 demonstrations in various situations. Two of them are used for training and the other six for validation.

As introduced in Sec. IV-A, since this task is simple, based on our prior information, we manually divide the whole demonstrations into two sub-skills based on the position of frame $\{2\}$. The former one is to move from the green box and reach the end of the magenta pipe, while the latter one is to start from the pipe end and move to the blue box. In Fig. 10a, the segmentation point is shown as a black dot.

By applying our method to each of the two sub-tasks, we present the solved frame weights in Fig. 10. For the first sub-task, Fig. 10b display the weights of frame $\{1\}$ and $\{2\}$. Similar to the simulated two-frame task introduced in Sec. V, the weight of frame $\{1\}$ goes from 1 to 0 along the trajectory, while the weight of frame $\{2\}$ increases from 0 to 1. Fig. 10c show the frame weights of $\{2\}$ and $\{3\}$ with respect to the second sub-task. Different from the weight of frame $\{2\}$ in the first sub-task, without the constraint at the beginning of the trajectory, the weight of frame $\{3\}$ directly increase to 1 without remaining 0 in the starting phase.

To generalize the movement in various situations, we first generate the trajectory in each sub-task, and then connect them together. From Fig. 11, we can observe the frame-weighted motion generation demonstrates good generalization capability, both in interpolation and extrapolation.

In a nutshell, the application of our proposed approach in this three-frame task indicates its ability to learn skills with multiple reference frames involved. For more complex tasks which are difficult to be manually divided, our method remains competitive with the help of automatic demonstration segmentation algorithms [17] [21] [22].

C. Three-frame task: robot experiments

Similar to the task shown in Fig. 10, we use a real robot experiment to further demonstrate the validity of our proposed method in multi-frame tasks. The setup is shown in Fig. 12. Two training demonstrations and calculated frame weights are displayed in Fig. 13. The generalization on the training and validation dataset is presented in Fig. 14. Since these results are similar to those shown in the last subsection, we do not illustrate them in depth here.

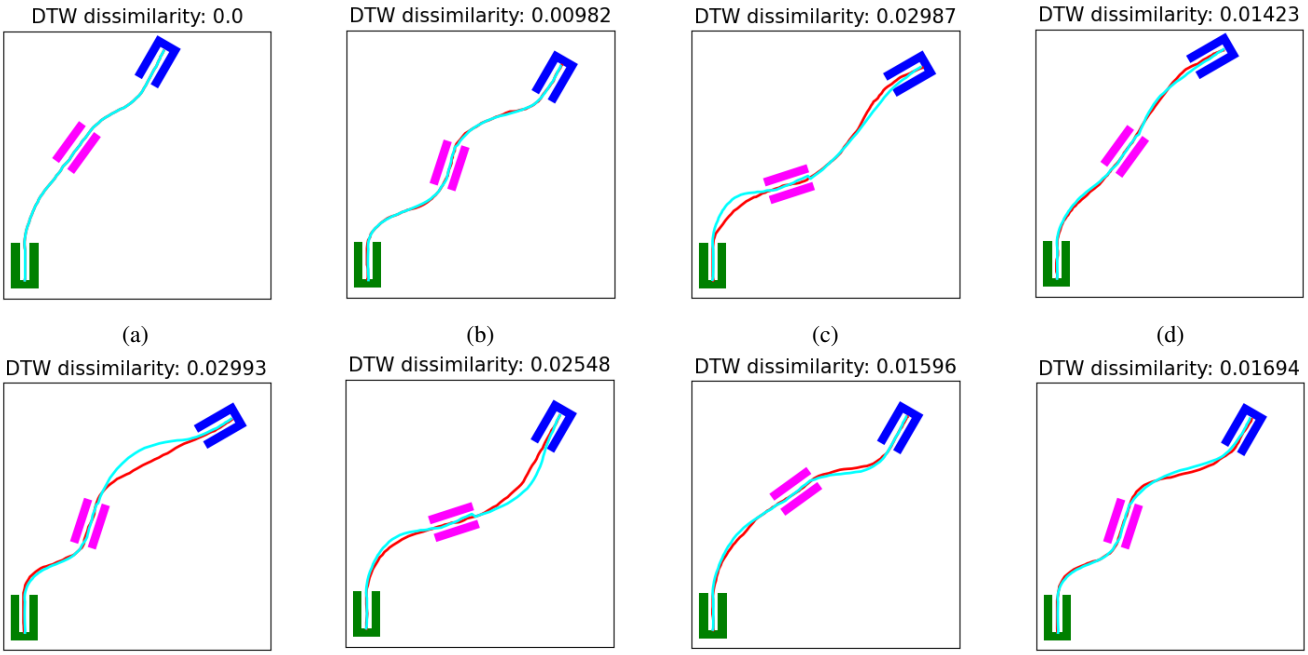


Fig. 11: (a) – (b) and (c) – (h) show generalizations in the training and validation dataset, respectively. The generalized trajectories and ground truths are shown in cyan and red. The DTW dissimilarity is displayed at the top of each sub-image. The smaller value indicates the better performance.

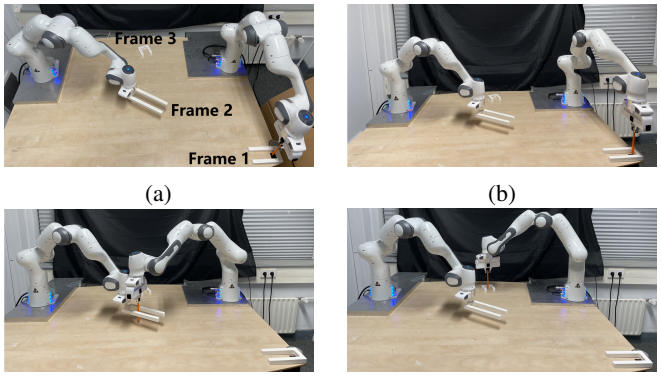


Fig. 12: The setup of the three-frame moving task. (a) The right robot moves the pen out of one U-shape box on the table (frame $\{1\}$), into the box mounted on the left robot (frame $\{2\}$), and finally into the other box on the table (frame $\{3\}$). (b) to (d) display the pen reaching each box.

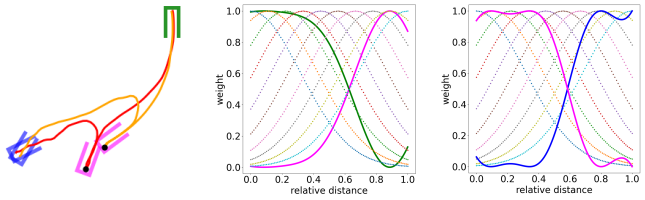


Fig. 13: (a) displays two demonstrations of the three-frame real robot experiment. The black dots indicates the segmentation of demonstrations. Three frames are attached to the green (frame $\{1\}$), magenta (frame $\{2\}$), and blue (frame $\{3\}$) U-shape boxes. (b) displays the weights of frame $\{1\}$ (green) and $\{2\}$ (magenta) on the first sub-trajectory. (c) displays the weights of frame $\{2\}$ (magenta) and $\{3\}$ (blue) on the second sub-trajectory.

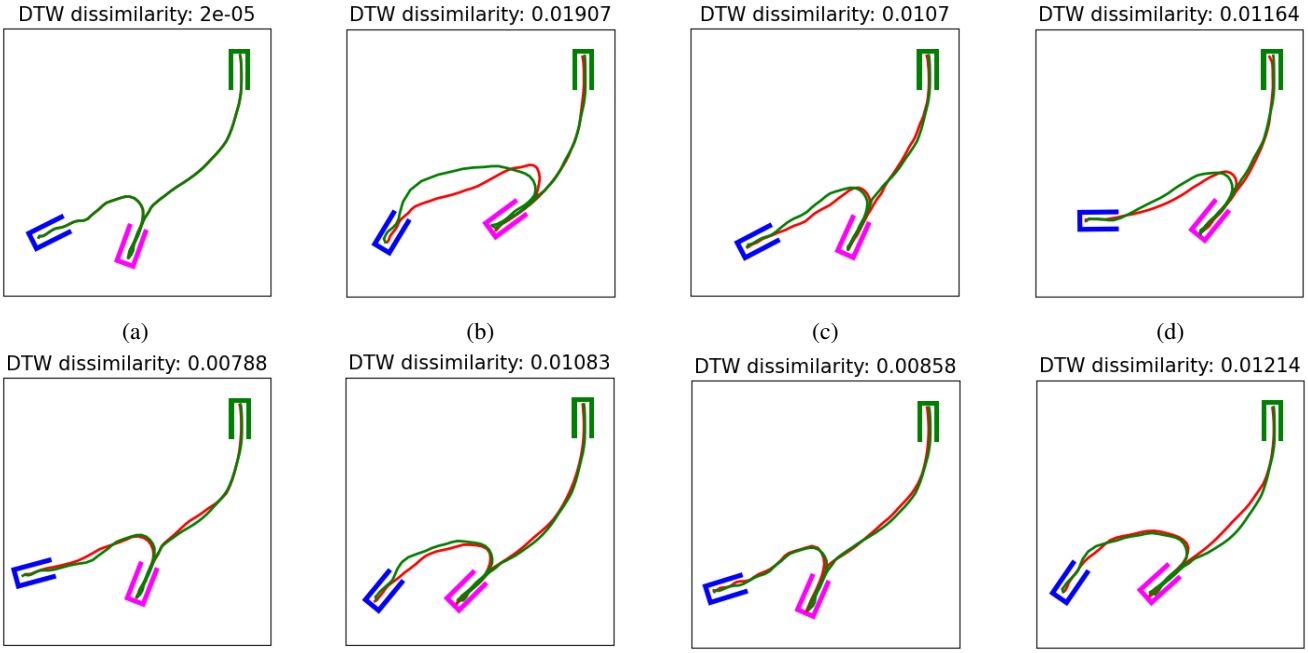


Fig. 14: (a) – (b) and (c) – (h) show generalizations in the training and validation dataset, respectively. The generalized trajectories and ground truths are shown in green and red. The DTW dissimilarity is displayed at the top of each sub-image. The smaller value indicates the better performance.