Optimizing Sequences of Probabilistic Manipulation Skills Learned from Demonstration — Supplementary Content —

Lukas Schwenkel
Institute for Systems Theory and Automatic Control, Stuttgart, Germany
Lukas.Schwenkel@ist.uni-stuttgart.de

Meng Guo, Mathias Bürger
Bosch Center for Artificial Intelligence (BCAI), Renningen, Germany
Meng.Guo2, Mathias.Buerger@de.bosch.com

3rd Conference on Robot Learning (CoRL 2019), Osaka, Japan.

1 Preliminaries

In this section we briefly introduce some preliminary results on using TP-GMMs and hidden semi-Markov models (HSMMs) for robot skill learning, and more importantly their adaptation to Riemannian Manifold.

1.1 TP-GMMs

The basic idea of LfD is to fit a prescribed skill model such as GMMs to a handful of demonstrations. It is assumed we are given M demonstrations, each of which contains T_m data points for a dataset of $N = \sum_{m=1}^{M} T_m$ total observations $\boldsymbol{\xi} = \{\boldsymbol{\xi}_t\}_{t=1}^{N}$, where $\boldsymbol{\xi}_t \in \mathbb{R}^d$ for sake of clarity. Also, we assume the same demonstrations are recorded from the perspective of P different coordinate systems TP (also called task parameters). One common way to obtain such data is to transform the demonstrations from global frame to frame $\mathsf{p} \in \mathsf{TP}$ by $\boldsymbol{\xi}_t^{(\mathsf{p})} = T^{-1}_{(\boldsymbol{b}_t^{(\mathsf{p})}, \boldsymbol{A}_t^{(\mathsf{p})})}(\boldsymbol{\xi}_t) = {\boldsymbol{A}_t^{(\mathsf{p})}}^{-1}(\boldsymbol{\xi}_t - \boldsymbol{b}_t^{(\mathsf{p})})$, where $\left(\boldsymbol{b}_t^{(\mathsf{p})}, \boldsymbol{A}_t^{(\mathsf{p})}\right)$ is the translation and rotation of frame p w.r.t. the world frame at time t, which we assume is available in this work. Then, a TP-GMM is described by the parameters $\{\pi_k, \{\boldsymbol{\mu}_k^{(\mathsf{p})}, \boldsymbol{\Sigma}_k^{(\mathsf{p})}\}_{\mathsf{p}\in\mathsf{TP}}\}_{k=1}^{K}$ where K represents the number of Gaussian components in the mixture model, π_k is the prior probability of each component, and $\boldsymbol{\mu}_k^{(\mathsf{p})}, \boldsymbol{\Sigma}_k^{(\mathsf{p})}$ are mean and covariance of the k-th component within frame p. Differently from standard GMM learning described in [1], the mixture model above can not be learned independently for each frame. Indeed, the mixing coefficients π_k are shared by all frames and the k-th component in frame p must map to the corresponding k-th component in the global frame. Expectation-Maximization (EM) in [2] is a well-established method to learn such models.

Once learned, the TP-GMM can be used during execution to reproduce a trajectory for the learned skill. Namely, given the observed frames $\{\boldsymbol{b}_t^{(\mathsf{p})}, \boldsymbol{A}_t^{(\mathsf{p})}\}_{\mathsf{p}\in\mathsf{TP}}$, the learned TP-GMM is converted into one single GMM with parameters $\{\pi_k, \hat{\boldsymbol{\mu}}_{t,k}, \hat{\boldsymbol{\Sigma}}_{t,k}\}_{k=1}^{K}$, by multiplying the affine-transformed Gaussian components across different frames, as follows

$$\left(\hat{\boldsymbol{\Sigma}}_{t,k}\right)^{-1} = \sum_{\mathsf{p}\in\mathsf{TP}} \left(\hat{\boldsymbol{\Sigma}}_{t,k}^{(\mathsf{p})}\right)^{-1}, \quad \hat{\boldsymbol{\mu}}_{t,k} = \hat{\boldsymbol{\Sigma}}_{t,k} \sum_{\mathsf{p}\in\mathsf{TP}} \left(\hat{\boldsymbol{\Sigma}}_{t,k}^{(\mathsf{p})}\right)^{-1} \hat{\boldsymbol{\mu}}_{t,k}^{(\mathsf{p})}, \tag{1}$$

where the parameters of the updated Gaussian at each frame p are computed as $\hat{\boldsymbol{\mu}}_{t,k}^{(\mathsf{p})} = \boldsymbol{A}_t^{(\mathsf{p})}\boldsymbol{\mu}_k^{(\mathsf{p})} + \boldsymbol{b}_t^{(\mathsf{p})}$ and $\hat{\boldsymbol{\Sigma}}_{t,k}^{(\mathsf{p})} = \boldsymbol{A}_t^{(\mathsf{p})}\boldsymbol{\Sigma}_k^{(\mathsf{p})}{\boldsymbol{A}_t^{(\mathsf{p})}}^{T}$. More details can be found in the review paper by [3].

1.2 HSMMs

Hidden semi-Markov Models (HSMMs) extend standard hidden Markov Models (HMMs) by embedding temporal information of the underlying stochastic process. That is, while in HMM the underlying hidden process is assumed to be Markov, i.e., the probability of transitioning to the next state depends only on the current state, in HSMM the state process is assumed semi-Markov. This means that a transition to the next state depends on the current state as well as on the elapsed time since the state was entered. HSMMs have been extensively applied in the past in speech synthesis, see [4]. Recently, they have been successfully applied, in combination with TP-GMMs, for robot skill encoding to learn spatio-temporal features of the demonstrations from [5]. More specifically, a task-parametrized HSMM model consists of the following parameters

$$\boldsymbol{\theta} = \left\{ \{a_{kh}\}_{h=1}^{K}, (\mu_k^D, \sigma_k^D), \{\pi_k, \{\boldsymbol{\mu}_k^{(\mathsf{p})}, \boldsymbol{\Sigma}_k^{(\mathsf{p})}\}_{\mathsf{p}\in\mathsf{TP}}\} \right\}_{k=1}^{K}, \tag{2}$$

where a_{kh} is the transition probability from state k to h; (μ_k^D, σ_k^D) describe the Gaussian distributions for the duration of state k, i.e., the probability of staying in state k for a certain number of consecutive steps; $\{\pi_k, \{\boldsymbol{\mu}_k^{(\mathsf{p})}, \boldsymbol{\Sigma}_k^{(\mathsf{p})}\}_{\mathsf{p}\in\mathsf{TP}}\}_{k=1}^{K}$ is the TP-GMM introduced earlier and, for each k, describe the emission probability, i.e., probability of observation, corresponding to state k. The prior π_k, however, describes in an HSMM only the probability distribution of the initial component at $t = 1$. The probability distribution of the components at subsequent time steps is determined via the underlying Semi-Markov Model. Note that in an HSMM each state corresponds to a Gaussian

component in the *associated* TP-GMM. In many cases, we want the structure of the Semi-Markov Model to be linear, which means the sequence of states is deterministic and only the duration in each state is probabilistic. We can fix this linear structure beforehand to speed up the training algorithm by setting $\pi_k = \delta_{1k}$ and $a_{kh} = \delta_{(k+1)h}$ with $\delta_{ij} = 0$ for $i \neq j$ and $\delta_{ij} = 1$ for $i = j$. In the following we assume that the structure is always linear, which means that we assume that each skill always follows the same segments and that we train for each skill a separate model such that no bifurcations are needed. Remember that these skills are very basic, like `grasp object`, `move object`, or `release object`. Hence, this assumption does not exclude complex assembling tasks it only requires the bifurcations to be on the skill level, e.g. after `grasp object` we can either execute `move object` or `release object`, but `grasp object` itself has only one way to execute it (move robot arm towards the object and close the gripper).

Furthermore, given a starting state k_0 and a desired goal state k_T at a desired time instant T, the most-likely sequence of states $\boldsymbol{k}^\star$ in the absence of observations can be determined with Viterbi's algorithm for HSMMs, as shown in [6]. Denote by $\boldsymbol{k}^\star = k_0 k_1 \cdots k_T$, where $k_t \in \{1, \cdots, K\}$. Thus, $\boldsymbol{k}^\star$ is the sequence of states that should be tracked during reproduction of the skill. For instance, a LQG-based minimal intervention control is used in [3] to design the controller that reproduce such sequence of Gaussian components.

Last but not least, we experienced that for robot manipulation tasks, we often face a non-Euclidean state space, e.g., rotation matrices $SO(3) \subset \mathbb{R}^{3\times3}$ or unit quaternions $\mathcal{S}^3 \subset \mathbb{R}^4$ are frequently used to describe orientations of objects or robot end effectors. Straightforward approach that projects the solution of the Euclidean computation back onto the manifold often suffers from low accuracy and can sometimes lead to disastrous consequences due to sign-flips as mentioned in [7]. Therefore, we adapt fully the aforementioned TP-HSMM formalism to Rimannian manifold. The detailed theory and derivations can be found in Sec.2 of this document and more in [8].

2 Riemannian Manifold

For robot manipulation tasks, we often face a non-Euclidean state space, e.g., rotation matrices $SO(3) \subset \mathbb{R}^{3\times3}$ or unit quaternions $\mathcal{S}^3 \subset \mathbb{R}^4$ are frequently used to describe orientations of objects or robot end effectors. Many basic operations such as computing distances, mean, or derivatives should be performed differently as in Euclidean space. Most recent work relies on a straightforward approach that projects the solution of the Euclidean computation back onto the manifold, see [9]. However, this approach often suffers from low accuracy and can sometimes lead to disastrous consequences due to sign-flips as mentioned in [7]. Therefore, we adapt fully the aforementioned TP-HSMM formalism to Rimannian manifold. The detailed theory and derivations can be found in [8]. We briefly summarize below the concepts essential for this work.

Consider a Riemannian manifold $\mathcal{M}$, one point $\boldsymbol{p} \in \mathcal{M}$ and its tangent space $\mathcal{T}_{\boldsymbol{p}}\mathcal{M}$ at $\boldsymbol{p}$. Note that $\mathcal{M}$ is equipped with a proper definition of inner product over its tangent space and distance on the manifold. The *exponential* map $\mathrm{Exp}_{\boldsymbol{p}} : \mathcal{T}_{\boldsymbol{p}}\mathcal{M} \to \mathcal{M}$ maps any point in the tangent space at point $\boldsymbol{p}$ to the manifold while preserving its distance to $\boldsymbol{p}$. Inversely, the *logarithmic* map $\mathrm{Log}_{\boldsymbol{p}} : \mathcal{M} \to \mathcal{T}_{\boldsymbol{p}}\mathcal{M}$ maps any point on the manifold to a point on the tangent space at point $\boldsymbol{p}$. Moreover, the generalization of a Gaussian normal distribution to Riemannian manifolds is given by

$$\mathcal{N}_{\mathcal{M}}(\boldsymbol{p}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) \sim \exp\left(-\frac{1}{2}\,\mathrm{Log}_{\boldsymbol{\mu}}(\boldsymbol{p})^\top \boldsymbol{\Sigma}^{-1}\,\mathrm{Log}_{\boldsymbol{\mu}}(\boldsymbol{p})\right), \tag{3}$$

where the mean $\boldsymbol{\mu}$ lies on the manifold and the covariance $\boldsymbol{\Sigma}$ lies in the tangent space $\mathcal{T}_{\boldsymbol{\mu}}\mathcal{M}$. Given a set of points within $\mathcal{M}$, the associated Gaussian distribution can not be analytically determined but only empirically via an iterative likelihood maximization algorithm as shown by [8]. Furthermore, given several Gaussians within $\mathcal{M}$, the resulting product can be computed similarly in a empirical way by projecting the means to a tangent space and parallel transporting the covariance matrices. These operations are useful for the EM algorithm and during reproduction as discussed in Section 1.1.

Lastly, a task parameter $(\boldsymbol{b},\ \boldsymbol{A})$ in the Riemannian context consists of the translation on the manifold $\boldsymbol{b} \in \mathcal{M}$ and rotation matrix in the tangent space $\mathcal{T}_{\boldsymbol{e}}\mathcal{M}$ at the manifold origin $\boldsymbol{e}$. As a result, the transformation between a observation $\boldsymbol{\xi}'$ in the local frame and the corresponding observation $\boldsymbol{\xi}$ in

translate $\mathsf{a}_{\mathtt{t1}}$		insert $\mathsf{a}_{\mathtt{is}}$	
$\mathsf{TP}_{\mathtt{t}}$	Success Rate	$\mathsf{TP}_{\mathtt{i}}$	Success Rate
$\{\mathsf{d}, \mathsf{r}\}$	0.08	$\{\mathsf{b}_1, \mathsf{r}\}$	0.05
$\{\mathsf{d}, \mathsf{r}, \mathsf{o}\}$	0.26	$\{\mathsf{b}_1, \mathsf{r}, \mathsf{o}\}$	0.44
$\{\mathsf{d}, \mathsf{o}\}$	0.37	$\{\mathsf{b}_1, \mathsf{o}\}$	0.71
$\{\mathsf{r}, \mathsf{o}\}$	0.99	$\{\mathsf{r}, \mathsf{o}\}$	0.99

Table 1: The choice of task parameters and associated success rate, for skills `translate` and `insert` used in the experiment. Note that r stands for the robot arm pose, o the object pose, b_1 the pose of the container to insert the object, and d the destination frame of the transition skill `translate`. Detailed descriptions are given in Sec.6 of the original paper.

the world frame is given by

$$\boldsymbol{\xi} = \mathrm{Exp}_{\boldsymbol{b}}\left(\boldsymbol{A}\,\mathrm{Log}_{\boldsymbol{e}}(\boldsymbol{\xi}')\right), \quad \boldsymbol{\xi}' = \mathrm{Exp}_{\boldsymbol{e}}\left(\boldsymbol{A}^{-1}\,\mathrm{Log}_{\boldsymbol{b}}(\boldsymbol{\xi})\right), \tag{4}$$

which are useful for converting observations between different frames.

3 Choice of Task Parameters

As emphasized in Sec.4.1 of the original paper, the success of both learning and reproducing skills with TP-GMMs heavily depends on a *good* choice of task parameters, which the user has to provide. As shown in Table 1, different choices of task parameters can result in significant changes in the performance. For example, a proper choice of task parameters for the skill `grasp peg` is a coordinate frame attached to the peg and one at the robot arm initial pose. This allows the learned trajectory to have a smooth start and more importantly adapt to new poses of the peg for successful grasping. As rule of thumb, attaching frames to all involved objects O_a and to the robot arm initial pose indexed by r as well as using the free task parameters F_a for transition skills, i.e. $\mathsf{TP}_\mathsf{a} = \mathsf{O}_\mathsf{a} \cup \mathsf{F}_\mathsf{a} \cup \{\mathsf{r}\}$, covers many cases. However, this is not always the best choice, since some objects might produce irrelevant task parameters, which not only increases the computation cost but can also decrease the performance of reproduction. A problem that arises with time-varying task parameters like an object pose is that the TP-HSMM only encodes how the task parameter influences the robot arm motion, but not how the robot arm motion affects the objects pose. For example, while executing the skill `move object` the trajectory of the robot arm in the frame attached to the object is only a single constant point, because the object follows every motion of the robot arm while it is grasped. Thus, the robot arm will follow the object during reproduction, i.e. stay where the object is, since the trajectory generation does not know that the robot arm can be moved freely without leaving the single point component in the object frame. In this case, it is better to not to use the object frame as task parameter.

In order to automate the choice of a proper set of task parameters $\mathsf{TP}_\mathsf{a} \subseteq \mathsf{O}_\mathsf{a} \cup \mathsf{F}_\mathsf{a} \cup \{\mathsf{r}\}$, we can validate a choice by computing its reproduction error. For this we need a ground truth, which is given as demonstration. Usually, the set of demonstrations D_a is rather small, such that we have to use the same set of demonstrations D_a for training and validation. This yields to the validation:

$$V(\mathsf{TP}_\mathsf{a}) = \sum_{m=1}^{M_\mathsf{a}} \sum_{t=1}^{T_m} \left\| \mathrm{Log}_{\boldsymbol{\xi}_t}(\hat{\boldsymbol{\xi}}_t) \right\|, \tag{5}$$

where $\hat{\boldsymbol{\xi}}_t$ is the trajectory retrieved from model $\boldsymbol{\theta}_\mathsf{a}(\mathsf{TP}_\mathsf{a})$ for the task parameters from demonstration D_m.

The number of involved objects for a skill is usually small, then we can train the model for all combinations of task parameters and validate each choice. If the number of objects is higher, the user has to preselect some promising choices of task parameters to reduce the computation time. The examples in Table 1 for the skills `translate` (object o to destination d) and `insert` (object o into box b) show that in both cases it is better to not use the object frame as task parameter due to the reasons described above. The choices of not using the robot initial arm position r or the destination d or box b are as to expect even worse.

Precondition Model $\gamma_{1,\mathtt{a}}$			
$\mathrm{TP}_\mathtt{a}$	o	d	r
o	–	(3.4e-2, 3.2e-2)	**(9.1e-3, 4.8e-5)**
d	(-1.2e-2, 2.8e-2)	–	(-3.4e-3, 2.7e-2)
r	**(-1.2e-2, 1.1e-5)**	(9.6e-2, 2.3e-2)	–

Effect Model $\gamma_{T,\mathtt{a}}$			
$\mathrm{TP}_\mathtt{a}$	o	d	r
o	(3.4e-2, 3.2e-2)	(3.5e-2, 3.3e-2)	(3.8e-2, 3.2e-2)
d	**(2.5e-5, 1.0e-5)**	(0.0, 1.0e-5)	**(7.2e-3, 4.9e-5)**
r	(9.6e-3, 2.4e-2)	(9.6e-2, 2.3e-2)	(1.0e-1, 2.4e-2)

Table 2: The learned task-parameterized precondition model $\gamma_{1,\mathtt{a}}$ from (3) of the original paper and effect model $\gamma_{T,\mathtt{a}}$ from (4) of the original paper of the translate skill $\mathtt{a}_{\mathtt{t1}}$, which moves object o to the destination frame d via the robot arm r. Due to limited space, only the mean and variance of the TP-Gs in the x-direction is shown here.

4 Training the Precondition and Prediction Model

The precondition and prediction models have the structure of Task Parameterized Gaussians. Since a TP-G has one component, we do not need to train them with an EM-algorithm, but we can directly compute the means and covariances. First, we clarify some notations. We denote the empirical mean and covariance on the manifold $\mathcal{M}$ of the points $X \subseteq \mathcal{M}$ with $\mathrm{mean}_\mathcal{M}(X)$, $\mathrm{cov}_\mathcal{M}(X)$ and the Gaussian product on the manifold $\mathcal{M}$ with $\prod_\mathcal{M}$. Further, we denote the mapping from the object pose $\boldsymbol{p}_\mathsf{o}$ of object $\mathsf{o} \in \mathsf{O_a}$ to the reference frame attached to it with $(\boldsymbol{b}(\boldsymbol{p}_\mathsf{o}), \boldsymbol{A}(\boldsymbol{p}_\mathsf{o}))$. The transformation (4) from world coordinates to the reference frame $(\boldsymbol{b}(\boldsymbol{p}_\mathsf{o}), \boldsymbol{A}(\boldsymbol{p}_\mathsf{o}))$ is denoted $\boldsymbol{\xi} = T_{\boldsymbol{p}_\mathsf{o}}(\boldsymbol{\xi}')$ and its inverse $\boldsymbol{\xi}' = T^{-1}_{\boldsymbol{p}_\mathsf{o}}(\boldsymbol{\xi})$ and so is the transformation of Riemannian Gaussians $\mathcal{N}_\mathcal{M}(\boldsymbol{\mu}, \boldsymbol{\Sigma}) = T_{\boldsymbol{p}_\mathsf{o}}(\mathcal{N}_\mathcal{M}(\boldsymbol{\mu}^{(\mathsf{o})}, \boldsymbol{\Sigma}^{(\mathsf{o})}))$. For simplification, we will use this notation also for transformations in the pose space $\mathcal{M}_p$ and not only in the observation space $\mathcal{M}_\xi$, although these are in fact different mappings.

Thus, we can compute the models from the demonstrations from (1) of the original paper as follows:
Initial-initial-precondition model, for each $\mathsf{o} \in \mathsf{O_a} \cup \mathsf{F_a}$ and each $\mathsf{p} \in \mathsf{O_a} \cup \mathsf{F_a} \cup \{\mathsf{r}, \mathsf{w}\}$ compute

$$\boldsymbol{\mu}^{(\mathsf{p})}_{1,\mathsf{o}} = \mathrm{mean}_{\mathcal{M}_p}\left(\left\{T^{-1}_{\boldsymbol{p}_{1,\mathsf{p}}}(\boldsymbol{p}_{1,\mathsf{o}})\middle|\mathsf{D}_m\right\}_{m=1}^{M_\mathsf{a}}\right), \quad \boldsymbol{\Sigma}^{(\mathsf{p})}_{1,\mathsf{o}} = \mathrm{cov}_{\mathcal{M}_p}\left(\left\{T^{-1}_{\boldsymbol{p}_{1,\mathsf{p}}}(\boldsymbol{p}_{1,\mathsf{o}})\middle|\mathsf{D}_m\right\}_{m=1}^{M_\mathsf{a}}\right).$$

Initial-final-prediction model, for each $\mathsf{o} \in \mathsf{O_a}$ and for each $\mathsf{p} \in \mathsf{TP_a}$ compute

$$\boldsymbol{\mu}^{(\mathsf{p})}_{T,\mathsf{o}} = \mathrm{mean}_{\mathcal{M}_p}\left(\left\{T^{-1}_{\boldsymbol{p}_{1,\mathsf{p}}}(\boldsymbol{p}_{T_m,\mathsf{o}})\middle|\mathsf{D}_m\right\}_{m=1}^{M_\mathsf{a}}\right), \quad \boldsymbol{\Sigma}^{(\mathsf{p})}_{T,\mathsf{o}} = \mathrm{cov}_{\mathcal{M}_p}\left(\left\{T^{-1}_{\boldsymbol{p}_{1,\mathsf{p}}}(\boldsymbol{p}_{T_m,\mathsf{o}})\middle|\mathsf{D}_m\right\}_{m=1}^{M_\mathsf{a}}\right).$$

For a given state $\boldsymbol{s}, \boldsymbol{p}_\mathsf{F}$, we can combine the Gaussians in the world frame in order to compute $\hat{\boldsymbol{\mu}}_{1,\mathsf{o}}, \hat{\boldsymbol{\Sigma}}_{1,\mathsf{o}}$ and $\hat{\boldsymbol{\mu}}_{T,\mathsf{o}}, \hat{\boldsymbol{\Sigma}}_{T,\mathsf{o}}$ as follows

$$\mathcal{N}(\hat{\boldsymbol{\mu}}_{1,\mathsf{o}}, \hat{\boldsymbol{\Sigma}}_{1,\mathsf{o}}) = \prod_{\mathsf{p}\in\mathsf{O_a}\cup\mathsf{F_a}\cup\{\mathsf{r},\mathsf{w}\}}{}_{\mathcal{M}_\mathsf{p}} T_{\boldsymbol{p}_\mathsf{p}}\left(\mathcal{N}(\boldsymbol{\mu}^{(\mathsf{p})}_{1,\mathsf{o}}, \boldsymbol{\Sigma}^{(\mathsf{p})}_{1,\mathsf{o}})\right) \tag{6}$$

$$\mathcal{N}(\hat{\boldsymbol{\mu}}_{T,\mathsf{o}}, \hat{\boldsymbol{\Sigma}}_{T,\mathsf{o}}) = \prod_{\mathsf{p}\in\mathsf{TP_a}}{}_{\mathcal{M}_\mathsf{p}} T_{\boldsymbol{p}_\mathsf{p}}\left(\mathcal{N}(\boldsymbol{\mu}^{(\mathsf{p})}_{T,\mathsf{o}}, \boldsymbol{\Sigma}^{(\mathsf{p})}_{T,\mathsf{o}})\right). \tag{7}$$

We suggest to choose the task parameters for the precondition model as all involved objects $\mathsf{O_a}$, the free task parameters $\mathsf{F_a}$ as well as the robot arm initial state r and even the world frame w to incorporate not only relative but also absolute conditions, i.e. $\mathsf{TP}_{1,\mathsf{a}} = \mathsf{O_a} \cup \mathsf{F_a} \cup \{\mathsf{r}, \mathsf{w}\}$. Thereby, we cover all possible relations in between two or more of the objects involved. Further, for the prediction we suggest to use the same set of task parameters as used in the TP-HSMM $\boldsymbol{\theta}_\mathsf{a}$, since these are the only ones influencing the reproduced trajectory of the robot arm and thus the only ones that have impact on the outcome of the skill execution, i.e. $\mathsf{TP}_{T,\mathsf{a}} = \mathsf{TP_a}$. For example, the

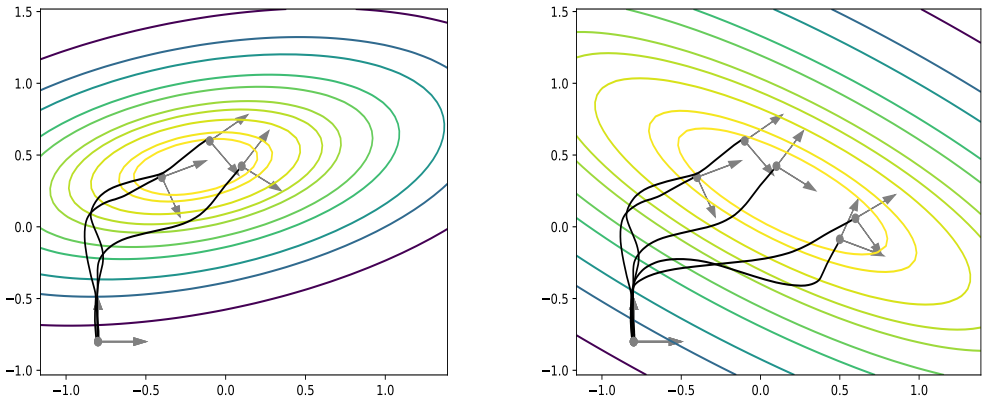

Figure 1: Example for the measure of confidence in $\mathbb{R}^2$ with three (left) and five (right) demonstrations. The task parameters are indicated by coordinates with gray arrows. The contour plots show the values of the confidence measure given different origins of the task parameter.

	SLSQP		Nelder-Mead	
Dim.	Time (s)	c_{a}	Time (s)	c_{a}
1	12.58	143.24	7.74	143.19
2	33.87	195.62	18.67	195.78
3	48.49	192.63	42.69	200.52
4	51.38	188.69	55.84	201.75
5	73.70	−81.68	61.82	202.75

Table 3: Computation time and the optimized confidence for the sequence of four skills used in the experiment under different solver and dimensions.

skill `move peg` uses as task parameters the robot initial arm pose and the goal pose for the motion, but not the peg itself. The precondition that the peg has to be grasped involves the peg and thus the task parameter set for the preconditions must include this task parameter that is not in TP_{a}. The prediction of the final peg pose, however, does only depend on the robot arm trajectory, which depends on the robot arm initial pose and on the desired goal pose, i.e. on TP_{a}.

Examples of the learned precondition and effect models for the skill `translate` used in the experiment section of the original paper are given in Table 2.

5 Numerical Examples

5.1 2D Example of Confidence Measure

A 2D example of the computed confidence measure is given in Figure 1. It shows that the confidence is higher close to the demonstrations and adaptive to the directions of variation in the demonstrated task parameters. Further, the area of highly confident task parameters can be enlarged by adding more demonstrations.

5.2 Computation Complexity for Optimization

Consider $\mathsf{d} \in \mathsf{F}_{\mathsf{a}}$ as the *destination* frame of the transition skill a. We optimize first in the d directions that have the largest variations demonstrated, i.e., over $\boldsymbol{\lambda} \in \mathbb{R}^d$ where $\boldsymbol{p}_{\mathsf{d}} = \hat{\boldsymbol{\mu}}_{1,\mathsf{d}} + \boldsymbol{V}\boldsymbol{\lambda}$ and $\boldsymbol{V} = [\boldsymbol{v}_1, \ldots, \boldsymbol{v}_d]$ are the eigenvectors of $\hat{\boldsymbol{\Sigma}}_{1,\mathsf{d}}$ corresponding to its d largest eigenvalues, with $(\hat{\boldsymbol{\mu}}_{1,\mathsf{d}}, \hat{\boldsymbol{\Sigma}}_{1,\mathsf{d}})$ being the mean and covariance of the destination frame based on the demonstrations of skill a. For example, when moving an object on a table the horizontal x and y directions should be

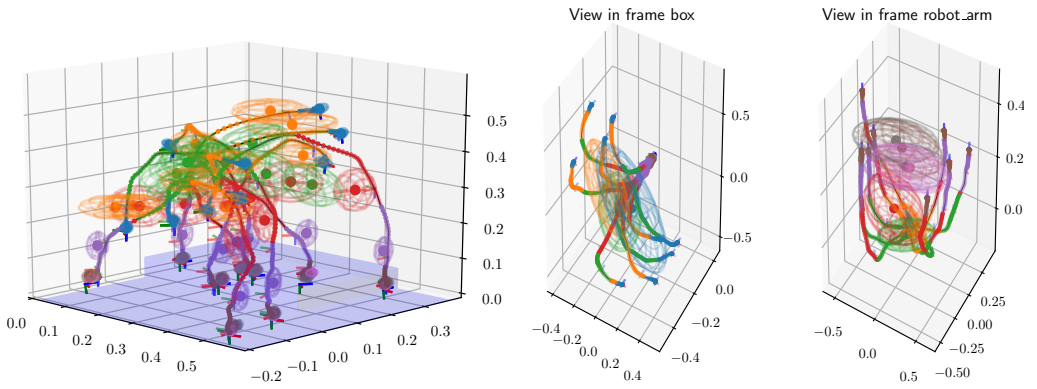

Figure 2: Demonstrations and the learned TP-GMMs in the global frame for skill "`grasp_top`". Note that each point on the trajectory is marked by same color as the Gaussian component it belongs to.

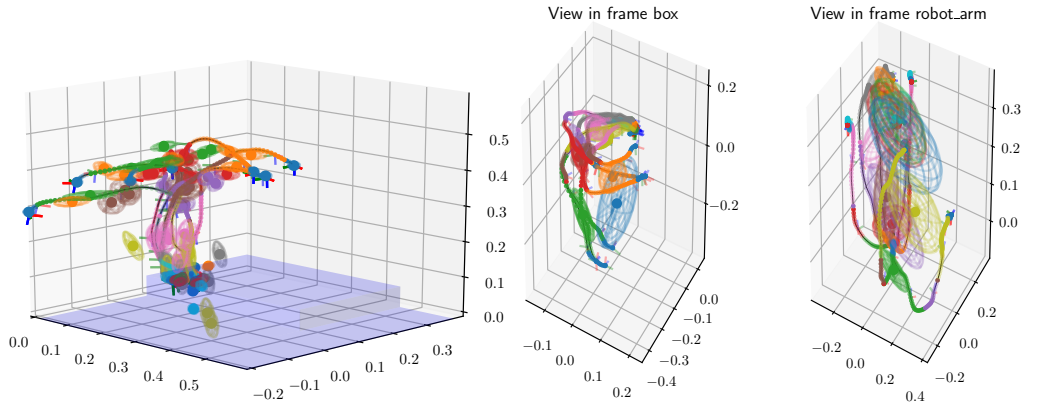

Figure 3: Demonstrations and the learned TP-GMMs in the global frame for skill "`grasp_side`". Note that each point on the trajectory is marked by same color as the Gaussian component it belongs to.

optimized, but the vertical z direction is fixed since we can not place the object in the air or inside the table and thus is unnecessary to optimize. An example of the optimized complexity under different solvers and dimensions is shown in Table 3.

5.3 Detailed Models Learned for the Experiments

In this section, we show for each skill used in the experiment section of the actual paper: actual demonstrations performed and the learned TP-HSMM models.

The demonstrations and the learned GMMs in the global frame for `grasp_top` skill $\mathtt{a}_{\mathtt{gt}}$ and the `translate` skill $\mathtt{a}_{\mathtt{tl}}$ are shown in Fig. 2.

5.4 Confidence Comparison for Different Scenarios

After learning the skill models above, given desired sequence $\mathbf{a}$ and the observed scenario $\boldsymbol{s}_0$, the confidence $c_{\mathbf{a}}(\cdot)$ can be maximized if it contains the transition skill $\mathtt{a}_{\mathtt{tl}}$. The criterion to choose $\mathbf{a}_1$ or $\mathbf{a}'_1$ is as follows: $\mathbf{a}'_1$ is chosen if the relative improvement (IR) defined by $(c_{\mathbf{a}'_1} - c_{\mathbf{a}_1})/(c_{\mathbf{a}_1} - c_{\min})$, where $c_{\min}$ is a predefined confidence lower bound, which is set to -10 here. The same rule applies to the choice of $\mathbf{a}_2$ and $\mathbf{a}'_2$. A comparison of confidences for a selection of initial object poses is shown in Table 4, where optimal choice for free TP destination $\mathtt{d}$ is shown along with the computation time. The distribution of IR over the *whole* workspace is shown in Figure 7. It can be seen that: (I) for tasks $\mathbf{a}_1$ and $\mathbf{a}'_1$, the difference in their confidences is mostly negligible as the skill

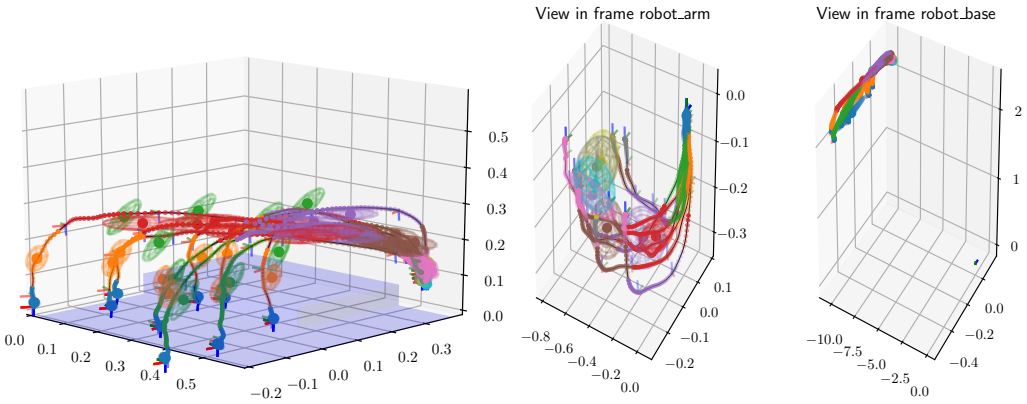

Figure 4: Demonstrations and the learned TP-GMMs in the global frame for skill "`drop`". Note that each point on the trajectory is marked by same color as the Gaussian component it belongs to.

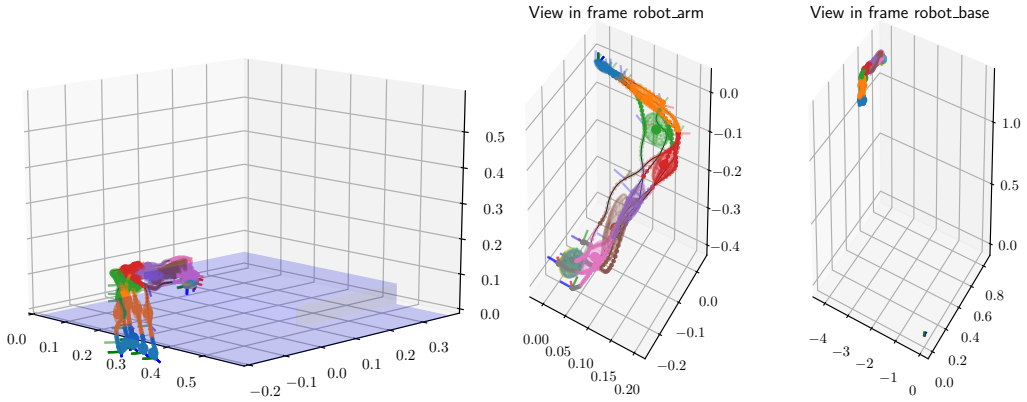

Figure 5: Demonstrations and the learned TP-GMMs in the global frame for skill "`insert`". Note that each point on the trajectory is marked by same color as the Gaussian component it belongs to.

`grasp_top` has quite high confidence across the workspace. However, it is worth noticing that close to top-left corner, $\mathbf{a}'_1$ is preferred over $\mathbf{a}_1$ because skill `drop` has never been demonstrated around that area, thus it is beneficial to translate the box to the center area where skill `drop` is more confident. (II) for tasks $\mathbf{a}_2$ and $\mathbf{a}'_2$, it is almost always beneficial to translate the object onto the platform first when the object is initially on the table, while this translation is unnecessary if the object is already on the platform. The main reason is that skill `grasp_side` is unsafe for the robot due to potential collision with the table, thus only demonstrated on the platform. Examples of execution trajectory of tasks $\mathbf{a}'_1$ and $\mathbf{a}'_2$ are shown in Fig. 8.

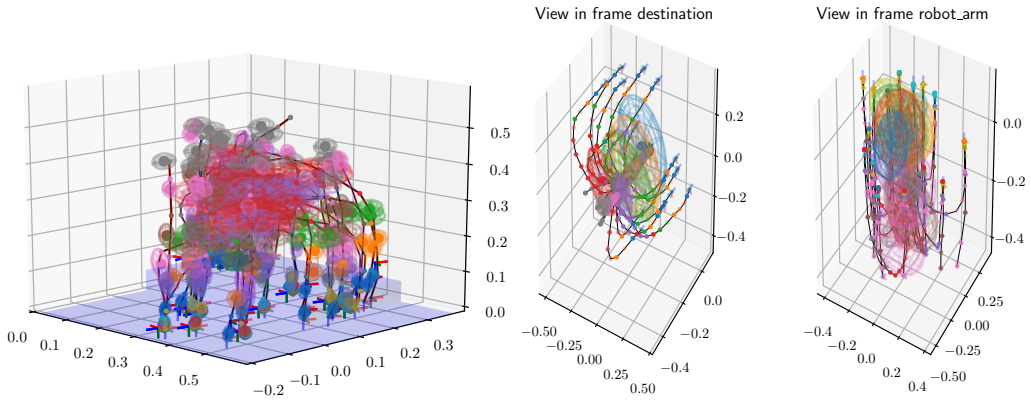

Figure 6: Demonstrations and the learned TP-GMMs in the global frame for skill "`translate`". Note that each point on the trajectory is marked by same color as the Gaussian component it belongs to.

$\boldsymbol{p}_{\text{box}}$	$c_{\mathbf{a}_2}$	$c_{\mathbf{a}'_2}$	$\boldsymbol{p}^{\star}_{\text{d}}$	$T_{\texttt{opt}}$ (s)	$c_{\mathbf{a}_1}$	$c_{\mathbf{a}'_1}$	$\boldsymbol{p}^{\star}_{\text{d}}$	$T_{\texttt{opt}}$ (s)
(0.5, 0.0, 0.03)	-580	**37**	(0.36, 0.27, 0.07)	13.6	**38**	37	(0.42, 0.12, 0.03)	21.7
(0.6, 0.1, 0.03)	-800	**32**	(0.37, 0.28, 0.07)	16.5	**32**	31	(0.40, 0.12, 0.03)	15.2
(0.6, 0.2, 0.03)	-800	**33**	(0.36, 0.28, 0.07)	15.8	**32**	30	(0.40, 0.12, 0.03)	21.3
(0.3, -0.08, 0.03)	-197	**20**	(0.36, 0.19, 0.07)	16.5	-1.2	**19**	(0.40, 0.1, 0.03)	12.2
(0.35, -0.0, 0.03)	-70	**28.5**	(0.37, 0.19, 0.07)	13.5	15.5	**28**	(0.44, 0.1, 0.03)	11.2
(0.3, 0.2, 0.07)	**46**	31	(0.36, 0.24, 0.07)	15.1	30	**31**	(0.41, 0.12, 0.03)	19.1
(0.4, 0.3, 0.07)	**41**	34	(0.36, 0.28, 0.07)	16.7	**31**	31	(0.41, 0.13, 0.03)	22.2
(0.3, 0.4, 0.07)	28	**32**	(0.36, 0.30, 0.07)	24.6	30	**31**	(0.42, 0.13, 0.03)	20.6

Table 4: Comparison of the confidence measure at different initial object poses, for both tasks with (i.e., $\mathbf{a}'_1$ and $\mathbf{a}'_2$) and without transition skills (i.e., $\mathbf{a}_1$ and $\mathbf{a}_2$).

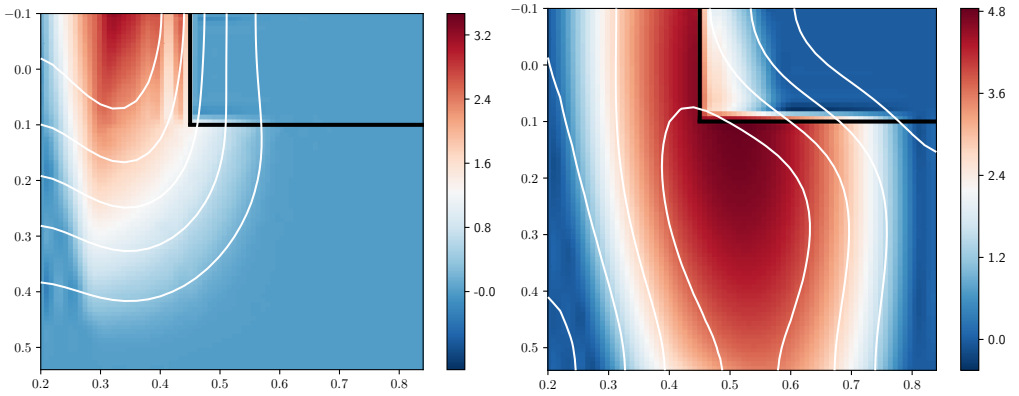

Figure 7: Heatmap of relative improvement over the workspace ($\mathbf{X}$, $\mathbf{Y}$-axis in m) when comparing $\mathbf{a}_1$ and $\mathbf{a}'_1$ (Left), and $\mathbf{a}_2$ and $\mathbf{a}'_2$ (Right). The area within the black lines is elevated by the platform.

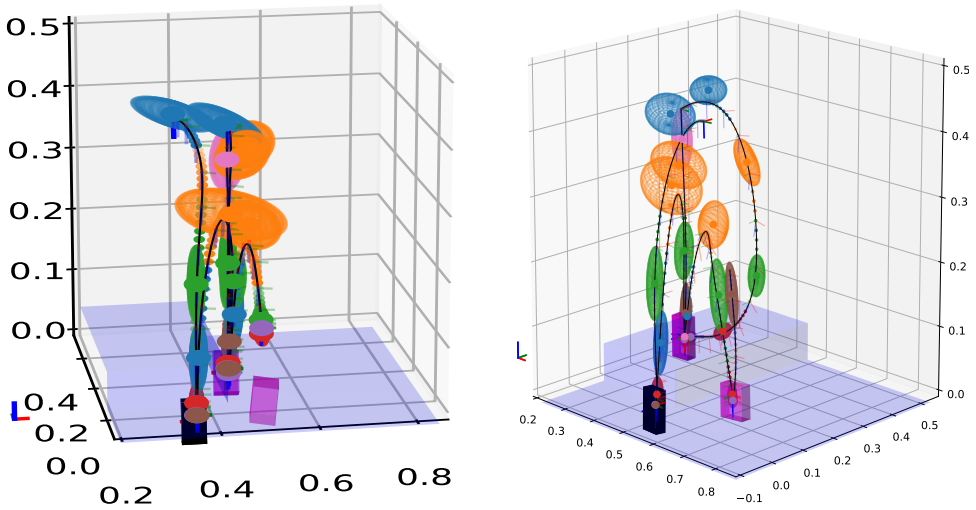

Figure 8: **Left**: One execution trajectory for task $\mathbf{a}'_1 = \mathtt{a}_{\mathtt{gt}}\mathtt{a}_{\mathtt{tl}}\mathtt{a}_{\mathtt{gt}}\mathtt{a}_{\mathtt{dp}}$. **Right**: One execution trajectory for task $\mathbf{a}'_2 = \mathtt{a}_{\mathtt{gt}}\mathtt{a}_{\mathtt{ts}}\mathtt{a}_{\mathtt{gs}}\mathtt{a}_{\mathtt{is}}$. Boxes in shaded magenta are the predicted intermediate poses during execution, while box in solid black is the initial pose.

References

[1] S. Niekum, S. Osentoski, G. Konidaris, S. Chitta, B. Marthi, and A. G. Barto. Learning grounded finite-state representations from unstructured demonstrations. *The International Journal of Robotics Research*, 34(2):131–157, 2015.

[2] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B*, 39(1):1–38, 1977.

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

[4] H. Zen, K. Tokuda, T. Masuko, T. Kobayasih, and T. Kitamura. A Hidden Semi-Markov Model-Based Speech Synthesis System. *IEICE Transactions on information and systems*, 90(5):825–834, 2007.

[5] A. K. Tanwani and S. Calinon. Learning Robot Manipulation Tasks with Task-Parameterized Hidden Semi-Markov Model. *IEEE Robotics and Automation Letters*, pages 1–8, 2016.

[6] S.-Z. Yu and H. Kobayashi. A hidden semi-Markov model with missing data and multiple observation sequences for mobility tracking. *Signal Processing*, 83(2):235–250, 2003.

[7] A. J. Hanson. Visualizing quaternions. In *ACM SIGGRAPH 2005 Courses*, SIGGRAPH '05, New York, NY, USA, 2005. ACM. doi:10.1145/1198555.1198701. URL `http://doi.acm.org/10.1145/1198555.1198701`.

[8] M. Zeestraten. Programming by demonstration on Riemannian manifolds. 2017. PhD thesis.

[9] M. Benosman and G. Le Vey. Control of flexible manipulators: A survey. *Robotica*, 22(5): 533545, 2004. doi:10.1017/S0263574703005642.