

Learning Task Parameterized Dynamic Movement Primitives using mixture of GMMs

Affan Pervez · Dongheui Lee

Abstract Task-parameterized skill learning aims at adaptive motion encoding to new situations. While existing approaches for task parameterized skill learning have demonstrated good adaptation within the demonstrated region, the extrapolation problem of task parameterized skills has not been investigated enough. In this work, with the aim of good adaptation not only within the demonstrated region but also outside of the region, we propose to combine a generative model with a Dynamic Movement Primitive (DMP) by formulating learning as a density estimation problem. Moreover, for efficient learning from relatively few demonstrations, we propose to augment training data with additional incomplete data. The proposed method is tested and compared with existing works in simulations and real robot experiments. Experimental results verified its generalization in the extrapolation region.

Keywords Programming by Demonstration · Dynamic Movement Primitives · Task Parameterized Movement

1 Introduction

Humans are very good at learning and reproducing complex tasks, but it is often tedious and cumbersome to program a robot for performing them. Programming by Demonstration (PbD) alleviates this problem, giving the possibility to teach a skill to a robot through demonstrations. The skill can be acquired from an expert in the corresponding field and removes the bottleneck for the teacher to have knowledge of robotics

A. Pervez, D. Lee
E-mail: {affan.pervez,dhlee}@tum.de
Authors are with Chair of Automatic Control Engineering,
Technische Universität München, 80333, Germany

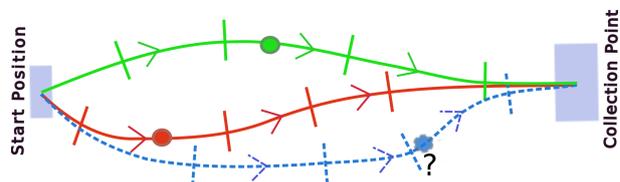


Fig. 1: This figure shows the illustration of a sweeping task. The position of the trash (colored circles) can be considered as the task parameter, governing variations in the demonstrations. For a new trash position (blue circle), which is away from the demonstrated region, the robot should be able to generate trajectory for moving trash to the collection point.

or programming. This also provides a great potential for industrial applications as PbD can significantly reduce the setting up time of an assembly line. Since PbD aims at speeding the setting up time, collecting a lot of demonstrations can be an expensive and time consuming task. Thus it is a desirable attribute to learn from as few demonstrations as possible. Moreover, since the demonstrations are finite, the learned controller should not only be able to generate motions within the demonstrated ranges, but also beyond them. There are skills in which multiple demonstrations can look very different due to the underlying task specific variations [18,27,28]. As an example, Figure 1 presents a sweeping task. In this task, the trash position can completely modify the trajectory of the broom, even for a fixed starting and collection point. For this task, the trash position can be interpreted as a *task parameter*, governing the variations in different demonstrations.

This paper utilizes these types of task specific demonstrations. Our approach combines dynamical systems [19, 26] and statistical machine learning techniques [7]. The existing works extending DMPs to include task parameters have used discriminative approaches for learning [10, 18, 27, 28]. Discriminative models are used for model-

ing the dependence of an unobserved variable y on an observed variable x . This is done by modeling the conditional probability distribution $P(y|x)$, which is then used for predicting y from x . On the contrary, we combine a generative approach with the DMP, as it can make use of incomplete/unlabeled training data. This results in an 1-Step learning procedure similar to [27]. Generative models encode the joint distribution of $P(x, y)$ of the variables of interest. The conditional distribution can later be inferred from the joint distribution i.e. $P(y|x) = P(x, y)/p(x)$ where $p(x)$ is obtained by marginalizing out y from $P(x, y)$. The aim of this work is to learn from very few demonstrations which implies sparsely distributed data. We solve the data sparsity problem by augmenting training data with additional incomplete data while poor local optima are avoided with Deterministic Annealing Expectation Maximization (DAEM).

The main contributions of this paper are:

- Instead of a discriminative model, we have used a generative model for modeling the forcing terms in a task parameterized DMP (Section 3).
- The local maxima problem during the likelihood maximization is resolved with DAEM (Section 3).
- We solve the data scarcity problem by using additional incomplete data and the Expectation Maximization (EM) algorithm [9] (Section 4). The detailed derivation of the incomplete data EM is also provided (Appendix).
- Through simulated and real robot experiments, we show that our approach requires very few demonstrations for learning and provides superior extrapolation capabilities when compared with the related works (Section 6).

2 Movement Primitives

2.1 Dynamic Movement Primitive

DMP is a way to learn motor actions [26]. It can encode discrete as well as rhythmic movements. We consider the DMP formulation presented in [19], as it overcomes the numerical problems which arises when changing the goal position in the original formulation [26]. A separate DMP is learned for each considered degree of freedom (DOF). A canonical system acts as a clock and for synchronization each DMP is driven by the common clock signal.

$$\tau \dot{s} = -\alpha_s s \quad (1)$$

The parameter s is usually initialized to one and it monotonically decays to zero, τ is the temporal scaling

factor while α_s determines the duration of the movement. From Equation (1), the time t and s are related as $s(t) = \exp(\frac{-\alpha_s t}{\tau})$. The canonical system drives the second order transformed system:

$$\tau \dot{v} = k(g - x) - dv - k(g - x_0)s + sk\mathcal{F}(s)$$

$$\tau \dot{x} = v$$

where g and x_0 are goal and start positions respectively, k acts like a spring constant while the damping term d is set such that the system is critically damped. The learning of forcing term $\mathcal{F}(s)$ allows arbitrarily com-

plex movements. $\mathcal{F}(s)$ is defined as $\frac{\sum_{i=1}^K \psi_i(s)\omega_i}{\sum_{i=1}^K \psi_i(s)}$ where

$\psi_i(s) = \exp(-h_i(s - c_i)^2)$ are Gaussian basis functions with spread h_i , centers c_i and adjustable weights ω_i . To encode a movement, we first register $x(t)$ and its first and second derivatives $v(t)$ and $\dot{v}(t)$ respectively at each time step $t = 0, \dots, T$. Then for a suitable value of τ , we integrate the canonical system and calculate the target value $\mathcal{F}_{tar}(s)$ for each time step.

$$\mathcal{F}_{tar}(s) = \frac{\dot{v}\tau - k(g - x) + dv + k(g - x_0)s}{sk}$$

Now learning is performed to minimize the error criterion $J = \sum_s (\mathcal{F}_{tar}(s) - \mathcal{F}(s))^2$ which is a linear regression problem and the weights ω_i are learned with weighted least squares.

2.2 DMP learning with a Gaussian Mixture Model

The forcing term $\mathcal{F}(s)$ can be encoded with a Gaussian Mixture Model (GMM) [2] or any other suitable function approximator [27]. When using a GMM, the manual specification of the meta parameters related to the basis functions (means and spread) is not required as the means and covariances of the GMM components are learned using EM. That is why the GMM based encoding also requires less number of components as compared with the number of basis functions. The number of GMM components can also be optimized by using an appropriate model selection criterion [22]. A GMM with K components is parameterized by $\theta_{(K)} = \{\pi_k, \mu_k, \Sigma_k\}_{k=1}^K$, where π_1, \dots, π_K are mixing coefficients with constraints $\pi_k > 0$ and $\sum_{k=1}^K \pi_k = 1$, μ_1, \dots, μ_K are means and $\Sigma_1, \dots, \Sigma_K$ are covariance matrices. The learning scheme is as follows. First a dataset is created

$$\mathbf{x} = \begin{pmatrix} s_1 & \dots & s_T \\ \mathcal{F}_{tar}(s_1) & \dots & \mathcal{F}_{tar}(s_T) \end{pmatrix} \quad (2)$$

Then a GMM is fitted to the data with EM [9]. Eigenvalues of covariance matrices are regularized to avoid

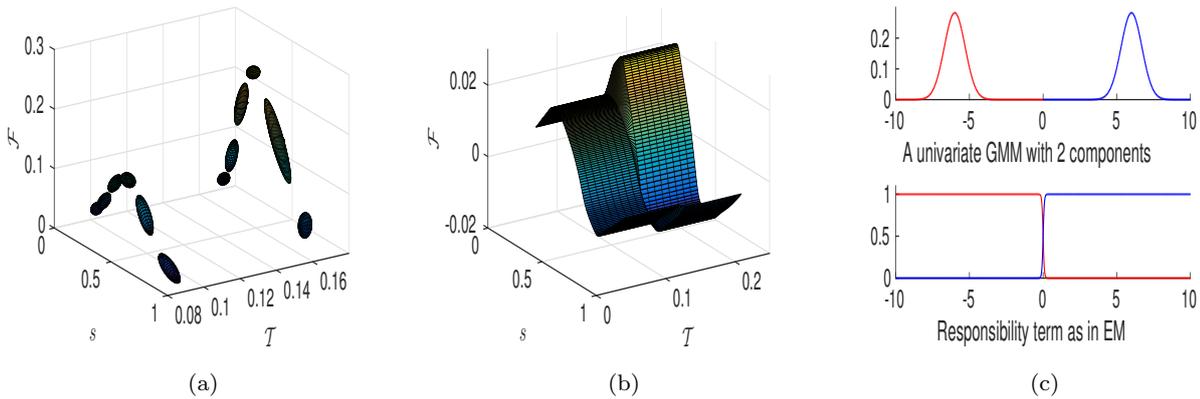


Fig. 2: TP-DMP learning using mixture of GMMs. (a) The mixture of GMMs, (b) the underlying regression surface and (c) the intuitive reasoning for such a response.

singularity during EM [5]. Now for retrieving $\mathcal{F}(s)$ for a given s value we can use Gaussian Mixture Regression (GMR) [4]. In GMR, input and output variables in each component are represented separately

$$\boldsymbol{\mu}_k = \begin{bmatrix} \boldsymbol{\mu}_k^I \\ \boldsymbol{\mu}_k^O \end{bmatrix}, \boldsymbol{\Sigma}_k = \begin{bmatrix} \boldsymbol{\Sigma}_k^I & \boldsymbol{\Sigma}_k^{IO} \\ \boldsymbol{\Sigma}_k^{OI} & \boldsymbol{\Sigma}_k^O \end{bmatrix}$$

For a given input variable \mathbf{x}^I , the expected value of \mathbf{x}^O is calculated as:

$$E(\mathbf{x}^O | \mathbf{x}^I) = \sum_{k=1}^K h_k \hat{\mathbf{x}}_k$$

$$\text{with } h_k = \frac{\pi_k \mathcal{N}(\mathbf{x}^I; \boldsymbol{\mu}_k^I, \boldsymbol{\Sigma}_k^I)}{\sum_{l=1}^K \pi_l \mathcal{N}(\mathbf{x}^I; \boldsymbol{\mu}_l^I, \boldsymbol{\Sigma}_l^I)}$$

$$\hat{\mathbf{x}}_k = \boldsymbol{\mu}_k^O + \boldsymbol{\Sigma}_k^{OI} (\boldsymbol{\Sigma}_k^I)^{-1} (\mathbf{x}^I - \boldsymbol{\mu}_k^I)$$

3 Task Parameterized-DMP

The DMP parameters can be separated into two types: 1. the shape parameters ω_i associated with the basis functions and 2. the DMP meta parameters which are all parameters other than the shape parameters, i.e. τ, g, K , etc. DMP presented in Section 2 does not consider external parameters \mathcal{T} , which are referred to as *task parameters* in this work (e.g. the trash position in Figure 1). Also the only input to a DMP is the clock signal. In the *Task Parameterized-DMP* (TP-DMP), we firstly want to learn from multiple demonstrations executed for different task parameters. Secondly for adapting the motion to a new task, the task parameters should also be passed as an input along with the clock signal. Following are the preprocessing steps that we consider in our TP-DMP framework:

1. Since a common clock (canonical system) drive all DMPs, we assign a common time duration to all demonstrations.

2. All demonstrations are linearly resampled to have an equal number of samples. A sample inbetween two data points is created by linear interpolation of the neighboring data points.

3.1 TP-DMP learning using mixture of GMMs

We consider learning as a density estimation problem where we want to learn the joint distribution of $(s, \mathcal{T}, \mathcal{F})$. Learning a single GMM over all demonstrations encoding $(s, \mathcal{T}, \mathcal{F})$ can suffer from curse of dimensionality in higher dimensional space. That is why we learn a separate GMM for each demonstration in lower dimensional space i.e. (s, \mathcal{F}) , by first creating the datasets as in Equation (2) and then applying EM as mentioned in section 2.2.

$$\boldsymbol{\mu}_{o,m} = \begin{pmatrix} \boldsymbol{\mu}_{o,m}^s \\ \boldsymbol{\mu}_{o,m}^{\mathcal{F}} \end{pmatrix}, \boldsymbol{\Sigma}_{o,m} = \begin{pmatrix} \boldsymbol{\Sigma}_{o,m}^{ss} & \boldsymbol{\Sigma}_{o,m}^{s\mathcal{F}} \\ \boldsymbol{\Sigma}_{o,m}^{\mathcal{F}s} & \boldsymbol{\Sigma}_{o,m}^{\mathcal{F}\mathcal{F}} \end{pmatrix}$$

Here the subscript o and m denote the indexes of demonstrations and components respectively while the terms s and \mathcal{F} in superscript denote the dimensions corresponding to s and \mathcal{F} respectively. Since the task parameters remain constant during a demonstration, we can simply concatenate their values in the learned means of the GMM components. The diagonal values corresponding to the task parameters in the covariances are not learned and are set to small value ϵ

$$\boldsymbol{\mu}_{o,m} = \begin{pmatrix} \boldsymbol{\mu}_{o,m}^s \\ \mathcal{T}_o \\ \boldsymbol{\mu}_{o,m}^{\mathcal{F}} \end{pmatrix}, \boldsymbol{\Sigma}_{o,m} = \begin{pmatrix} \boldsymbol{\Sigma}_{o,m}^{ss} & 0 & \dots & 0 & \boldsymbol{\Sigma}_{o,m}^{s\mathcal{F}} \\ 0 & \epsilon & \ddots & & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & & & \ddots & \epsilon & 0 \\ \boldsymbol{\Sigma}_{o,m}^{\mathcal{F}s} & 0 & \dots & 0 & \boldsymbol{\Sigma}_{o,m}^{\mathcal{F}\mathcal{F}} \end{pmatrix}.$$

If the task parameters are varying during the demonstrations, the GMMs can simply be learned over all of the variables, i.e. $(s, \mathcal{T}, \mathcal{F})$, and we will have unconstrained covariances. In fact it is the fixed task parameters for each demonstration that makes the learning challenging. The idea of combining separately learned HMMs has also been introduced in [13] but in our approach we apply an additional EM cycle over the separately learned GMMs for achieving task specific generalization. We mix the separately learned GMMs to achieve generalization for novel task parameter values. Similar to the mixing coefficients π which represent the weights of the components within a GMM, we introduce a mixing coefficient ϕ representing the mixing weight of each GMM, having the same constraints as that of π , i.e. $\pi_i > 0$ and $\sum_{i=1}^K \pi_i = 1$. With these settings we apply EM to learn a mixture of GMMs. We do not update the mixing weights and the means of the components within each GMM as they are important for preserving the local behavior of each demonstration. As EM maximizes the likelihood locally, it can converge to a local maxima. To overcome local maxima problem we use DAEM [29]. DAEM has a temperature parameter β which has a small value at the beginning and it gradually increases to one. It uses the ideas from statistical mechanics, by applying the concept of maximum entropy. The objective function is considered as the thermodynamic free energy, which is regulated by the temperature. EM is applied deterministically for each temperature value and the estimated parameters become initialization for the next temperature value. At lower temperature values DAEM suppresses poor local optima and increases the likelihood of convergence to the global maximum. For M demonstrations with T data points in each demonstration, we first create a single dataset containing all demonstrations (define $N = TM$) and then apply the DAEM for learning mixture of GMMs, whose update equations can be written as:

E-step:

$$p_{i,o,m} = (\phi_o^t \pi_{o,m} \mathcal{N}(\mathbf{x}_i; \boldsymbol{\mu}_{o,m}^t, \boldsymbol{\Sigma}_{o,m}^t))^\beta$$

$$b_{i,o} = \frac{\sum_{l=1}^K p_{i,o,l}}{\sum_{r=1}^M \sum_{l=1}^K p_{i,r,l}}, \quad q_{i,o,m} = \frac{p_{i,o,m}}{\sum_{r=1}^M \sum_{l=1}^K p_{i,r,l}}$$

M-step:

$$\phi_o^{t+1} = \frac{\sum_{i=1}^N b_{i,o}}{N},$$

$$\boldsymbol{\Sigma}_{o,m}^{t+1} = \frac{\sum_{i=1}^N q_{i,o,m} (\mathbf{x}_i - \boldsymbol{\mu}_{o,m}) (\mathbf{x}_i - \boldsymbol{\mu}_{o,m})^\top}{\sum_{i=1}^N q_{i,o,m}}$$

where $b_{i,o}$ represents the responsibility that the o^{th} GMM takes for explaining the i^{th} data point while $q_{i,o,m}$ represents the responsibility that the m^{th} component of the o^{th} GMM takes for explaining the i^{th} data point. It is a common practice to apply regularization on learned covariances for avoiding singularities during EM. As a regularization measure, if any of the eigenvalues of the learned covariances becomes lower than a predefined threshold ϵ , then we reset it to ϵ . We apply the described approach to the mixture of GMMs learned for encoding the forcing terms of a DMP, with variations along a scalar task parameter. The learned mixture of GMMs, which is also the estimated density function, is shown in Figure 2a. The GMMs remain unchanged. The regression for the mixture of GMMs is calculated as:

$$E(\mathbf{x}^O | \mathbf{x}^I) = \sum_{r=1}^M \sum_{l=1}^K v_{r,l} \hat{\mathbf{x}}_{r,l}$$

$$\text{with } v_{o,m} = \frac{\phi_o \pi_{o,m} \mathcal{N}(\mathbf{x}^I; \boldsymbol{\mu}_{o,m}^I, \boldsymbol{\Sigma}_{o,m}^I)}{\sum_{r=1}^M \sum_{l=1}^K \phi_r \pi_{r,l} \mathcal{N}(\mathbf{x}^I; \boldsymbol{\mu}_{r,l}^I, \boldsymbol{\Sigma}_{r,l}^I)}$$

$$\hat{\mathbf{x}}_{o,m} = \boldsymbol{\mu}_{o,m}^O + \boldsymbol{\Sigma}_{o,m}^{OI} (\boldsymbol{\Sigma}_{o,m}^I)^{-1} (\mathbf{x}^I - \boldsymbol{\mu}_{o,m}^I).$$

For evaluation, we generate linearly spaced samples of \mathcal{T} and s within the demonstrated ranges and use GMR to predict the value of \mathcal{F} for each sample, i.e. $\{s \times \mathcal{T}\} \mapsto \mathcal{F}$. The surface plot of this data can be visualized in Figure 2b. We can see that it has a step along the task parameter. This shows that as we have sparse data in the task space (only two trajectories), each GMM in the mixture kept concentrated at regions of demonstrations. Due to data sparsity, the density estimate is overfitted in its current form. The reason for the step in surface plot can be explained by Figure 2c. If we have well separated Gaussians (Figure 2c-top) then their activation functions, calculated as the responsibility term in EM, switches in a very narrow region (Figure 2c-bottom). The same phenomenon occurred in the regression surface where regions close to a GMM are mostly influenced by it. As we move away, the activation transits sharply to a nearby GMM. This also means that this model is overfitted and can only be useful for exact reproduction of task parameter values in training dataset and it fails to generalize for novel task

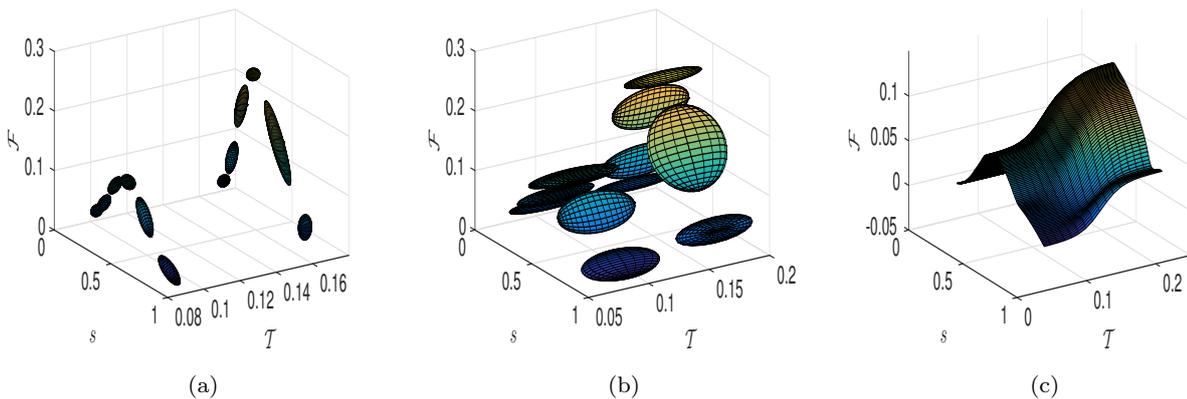


Fig. 3: Result of manually setting the variance along task variable. (a) The mixture of GMMs, (b) the mixture of GMMs after manually setting the variance along task variable and (c) the underlying regression surface.

parameter values. Instead of using EM, a simple trick to avoid overfitting problem could have been to manually specify a reasonable value of variance ϵ for the task variables, as in the LWR based approaches [28]. This is analogous of applying a regularization term along task variables. Although this can provide interpolation behavior, it fails to extrapolate beyond the demonstrated regions, as the underlying regression surface changes its behavior beyond the demonstrated interval as shown in Figure 3.

4 Generalizing from incomplete data via the EM

Due to the few demonstrations, the challenge of data sparsity is posed in task space. We show that the data sparsity problem can be solved by augmenting the training data with the additional incomplete data spanning the input space. This can subsequently be used for getting a better estimate of the underlying distribution and thus improving the generalization behavior of the already learned GMMs. Since the mixture of GMMs is a generative model, it can benefit from incomplete data [3].

4.1 Defining input data distribution

For a task parameterized DMP, the input variables are (s, \mathcal{T}) for which we want to predict the value of output \mathcal{F} . Without loss of generality, we can assume that the input variables are independent of each other but conditionally dependent for a given output value, as shown in Figure 4. We first separately model the distribution of each input variable. Among the input variables, the clock signal s is generated by an exponen-

tially decaying function (canonical system) and has uneven distribution of samples in different regions. Since a GMM can model any arbitrarily complex density function we model the distribution of s by fitting a univariate GMM with W components (through EM) to its samples $\theta_{(W)}^s = \{\pi_w^s, \mu_w^s, (\sigma_w^s)^2\}_{w=1}^W$. The same procedure cannot be used for the very limited samples of task parameters \mathcal{T} , which are equal to the number of demonstrations. For simplicity, we assume each dimension of task variables to follow a univariate normal distribution, i.e. for the d^{th} dimension of $\mathcal{T} = [\mathcal{T}^1 \dots \mathcal{T}^d \dots \mathcal{T}^D]^\top$, $\mathcal{T}^d \sim \mathcal{N}(\mu_d, \sigma_d^2)$ where $\mu_d = \frac{\mathcal{T}_1^d + \mathcal{T}_2^d + \dots + \mathcal{T}_M^d}{M}$ and $\sigma_d^2 = \frac{1}{M-1} \sum_{i=1}^M (\mathcal{T}_i^d - \mu_d)^2$.

Since the inputs are provided in a regression problem, we refer to them as the observable variables. Similarly, the output variables that need to be predicted are termed as missing variables. Using the assumption of independence, the resultant distribution of the input variables is defined by concatenating all the distribution learned separately to form a multivariate GMM with W components.

$$\theta_{(W)}^{obs} = \{\pi_w^{obs}, \mu_w^{obs}, \Sigma_w^{obs}\}_{w=1}^W \text{ with } \pi_w^{obs} = \pi_w^s,$$

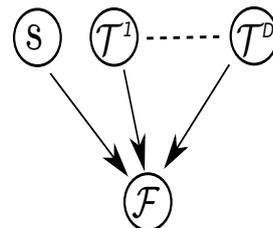


Fig. 4: Graphical model illustrating dependence of input and output variables.

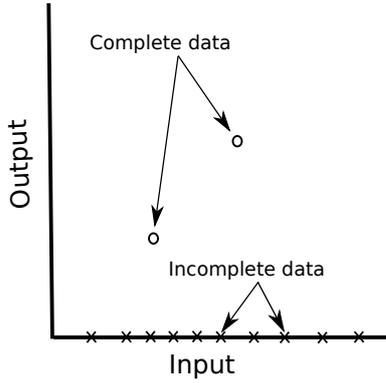


Fig. 5: If we have data points whose certain dimensions are missing then they can still be used when applying EM for fitting a GMM. For instance, the output value of the data plotted on the input axis is missing. For this incomplete data, the expected values of the missing output values are also estimated within EM.

$$\boldsymbol{\mu}_w^{obs} = \begin{bmatrix} \mu_w^s \\ \mu_1 \\ \vdots \\ \mu_D \end{bmatrix} \text{ and } \boldsymbol{\Sigma}_w^{obs} = \begin{bmatrix} (\sigma_w^s)^2 & 0 & \dots & 0 \\ 0 & \sigma_1^2 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \dots & 0 & \sigma_D^2 \end{bmatrix}.$$

As the output dimension is not considered for now, we term the input data distribution as Incomplete Data GMM (IDGMM).

4.2 Incorporating Incomplete data via EM

Figure 5 shows a dataset with a mixture of complete and incomplete data. The incomplete data still provides useful information when applying EM for fitting a GMM [9]. As mentioned earlier, the regression using mixture of GMMs encountered problem due to the data sparsity in task space. What would be the effect of filling regions inbetween GMMs with incomplete data, i.e. without the outputs \mathcal{F} ? The EM applied with additional incomplete samples provide smooth activation of responsibilities when switching from one GMM to a nearby GMM. Since we treat learning as a density estimation problem, the amount of incomplete data required to fill the empty regions can increase drastically with the increase in the dimensions of task parameters (curse of dimensionality). The computational burden of EM also increases with the increase in size of training data. To avoid these problems, we instead directly use the IDGMM.

To use the IDGMM, a weighting parameter analogous to the number of data points represented by the IDGMM has to be specified by the user. Our training dataset consists of N data points. The weighting parameter is also set equal to N and thus each IDGMM

component represent $\boldsymbol{\pi}_w^{obs} \times N$ data points. It has to be noted that the EM applied with this additional data still provides a maximum likelihood estimate of the model parameters as the IDGMM is defined on data and not on the parameters of the model. Now, for benefiting from this incomplete data, we use the current mixture of GMMs for calculating the Expectation of incomplete terms appearing in likelihood maximization [12]. It turns out that, for a data point \mathbf{x}_i with observable (input dimensions) and missing (output dimensions) parts $\begin{bmatrix} \mathbf{x}_i^{obs} \\ \mathbf{x}_i^{miss} \end{bmatrix}$, we have to calculate three expectations [12], i.e. $E[z_{i,k} | \mathbf{x}_i^{obs}, \boldsymbol{\theta}_t]$, $E[z_{i,k}, \mathbf{x}_i^{miss} | \mathbf{x}_i^{obs}, \boldsymbol{\theta}_t]$ and $E[z_{i,k}, \mathbf{x}_i^{miss} \mathbf{x}_i^{miss \top} | \mathbf{x}_i^{obs}, \boldsymbol{\theta}_t]$, where $z_{i,k}$ is an indicator variable defining association of \mathbf{x}_i to the k_{th} cluster. These expectations can be directly used in M-step [12].

In the M-step, the value of $d_{i,o,m}^{t+1}$ calculated only on the observable dimensions can be directly used for updating ϕ_o^{t+1} .

E-step:

$$p_{i,o,m}^{t+1} = (\phi_o^t \boldsymbol{\pi}_{o,m} \mathcal{N}(\mathbf{x}_i; \boldsymbol{\mu}_{o,m}^t, \boldsymbol{\Sigma}_{o,m}^t))^\beta$$

$$c_{w,o,m}^{t+1} = (\phi_o^t \boldsymbol{\pi}_{o,m} \mathcal{N}(\boldsymbol{\mu}_w^{obs}; \boldsymbol{\mu}_{o,m}^{obs}, \boldsymbol{\Sigma}_w^{obs} + \boldsymbol{\Sigma}_{o,m}^{obs}))^\beta$$

$$b_{i,o}^{t+1} = \frac{\sum_{l=1}^K p_{i,o,l}^{t+1}}{\sum_{r=1}^M \sum_{l=1}^K p_{i,r,l}^{t+1}}, \quad q_{i,o,m}^{t+1} = \frac{p_{i,o,m}^{t+1}}{\sum_{r=1}^M \sum_{l=1}^K p_{i,r,l}^{t+1}}$$

$$d_{w,o,m}^{t+1} = \frac{c_{w,o,m}^{t+1}}{\sum_{r=1}^M \sum_{l=1}^K c_{w,r,l}^{t+1}} \times \boldsymbol{\pi}_w^{obs} \times N$$

M-step:

$$\phi_o^{t+1} = \frac{\sum_{i=1}^N b_{i,o}^{t+1} + \sum_{w=1}^W \sum_{l=1}^K d_{w,o,l}^{t+1}}{2N},$$

$$\boldsymbol{\Sigma}_{o,m}^{t+1} = \frac{\sum_{i=1}^N q_{i,o,m}^{t+1} (\mathbf{x}_i - \boldsymbol{\mu}_{o,m}) (\mathbf{x}_i - \boldsymbol{\mu}_{o,m})^\top + \sum_{w=1}^W \mathbf{A}_{w,o,m}}{\sum_{i=1}^N q_{i,o,m}^{t+1} + \sum_{w=1}^W d_{w,o,m}^{t+1}}$$

where

$$\boldsymbol{\mu}_{o,m} = \begin{bmatrix} \boldsymbol{\mu}_{o,m}^{obs} \\ \boldsymbol{\mu}_{o,m}^{miss} \end{bmatrix}, \quad \boldsymbol{\Sigma}_{o,m} = \begin{bmatrix} \boldsymbol{\Sigma}_{o,m}^{obs} & \boldsymbol{\Sigma}_{o,m}^{obs,miss} \\ \boldsymbol{\Sigma}_{o,m}^{miss,obs} & \boldsymbol{\Sigma}_{o,m}^{miss} \end{bmatrix}$$

$$\mathbf{A}_{w,o,m} = d_{w,o,m}^{t+1} \begin{bmatrix} \mathbf{X}_w^{obs} - \boldsymbol{\mu}_{o,m}^{obs} \\ \mathbf{X}_w^{miss} - \boldsymbol{\mu}_{o,m}^{miss} \end{bmatrix} \begin{bmatrix} \mathbf{X}_w^{obs} - \boldsymbol{\mu}_{o,m}^{obs} \\ \mathbf{X}_w^{miss} - \boldsymbol{\mu}_{o,m}^{miss} \end{bmatrix}^\top$$

$$= \begin{bmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} \\ \mathbf{A}_{21} & \mathbf{A}_{22} \end{bmatrix}$$

with

$$\begin{aligned} \mathbf{A}_{11} &= d_{w,o,m}^{t+1} \left(\boldsymbol{\Sigma}_w^{obs} + (\boldsymbol{\mu}_w^{obs} - \boldsymbol{\mu}_{o,m}^{obs})(\boldsymbol{\mu}_w^{obs} - \boldsymbol{\mu}_{o,m}^{obs})^\top \right) \\ \mathbf{A}_{21} &= d_{w,o,m}^{t+1} \boldsymbol{\Sigma}_{o,m}^{miss.obs} (\boldsymbol{\Sigma}_{o,m}^{obs})^{-1} \mathbf{A}_{w,o,m} \\ \mathbf{A}_{12} &= \mathbf{A}_{21}^\top \\ \mathbf{A}_{22} &= d_{w,o,m}^{t+1} \boldsymbol{\Sigma}_{o,m}^{miss} + d_{w,o,m}^{t+1} \boldsymbol{\Sigma}_{o,m}^{miss.obs} \\ &\quad (\boldsymbol{\Sigma}_{o,m}^{obs})^{-1} \mathbf{A}_{w,o,m} (\boldsymbol{\Sigma}_{o,m}^{obs})^{-1} \boldsymbol{\Sigma}_{o,m}^{miss.obs}^\top \\ &\quad - d_{w,o,m}^{t+1} \boldsymbol{\Sigma}_{o,m}^{miss.obs} (\boldsymbol{\Sigma}_{o,m}^{obs})^{-1} (\boldsymbol{\Sigma}_{o,m}^{miss.obs})^\top \end{aligned}$$

where $\mathbf{A}_{w,o,m} = \boldsymbol{\Sigma}_w^{obs} + (\boldsymbol{\mu}_w^{obs} - \boldsymbol{\mu}_{o,m}^{obs})(\boldsymbol{\mu}_w^{obs} - \boldsymbol{\mu}_{o,m}^{obs})^\top$. Since we do not update the means, the additional data cannot pull the already learned GMMs away from the demonstrated regions, the components cannot get to a saddle point during DAEM and it also results in a fast rate of convergence for EM.

One may wonder about the result of fitting a single GMM instead of using the mixture of GMMs. As the data is concentrated at discrete regions of input space (task parameters), the components of a single GMM will also get attracted to same regions as by the mixture of GMMs. An appropriate regularization term must be used for a single GMM, to avoid singularity issues of covariance matrices, as the data concentrated at discrete regions in higher dimensional space. The reason for using the mixture of GMMs instead of a single GMM is that now we optimize the weight ϕ of each GMM separately, without disturbing the weights π within each GMM, as they are important for preserving the local behavior of each GMM. This also results in a smaller number of parameters update during the second EM cycle, in contrast to updating the mixing weights π of a single GMM.

4.3 Computational complexity

The computational complexity of our approach during motion execution is $\mathcal{O}(n)$ for the number of DMPs, the number of demonstrations and the number of GMM components. Since GMR involves matrix inversion over input variables, the computational complexity is $\mathcal{O}(n^3)$ for the dimensions of task parameters. For the special case of fixed task parameters throughout the trajectory, one can find conditional GMMs for the fixed task parameters. This makes the computational complexity irrelevant of task parameters, i.e. for the fixed task parameters \mathcal{T} , the conditional parameters for the regres-

sion are calculated as:

$$\begin{aligned} \hat{\phi}_o \hat{\pi}_{o,m} &= \frac{\phi_o \pi_{o,m} \mathcal{N}(\mathcal{T}; \boldsymbol{\mu}_{o,m}^\mathcal{T}, \boldsymbol{\Sigma}_{o,m}^\mathcal{T})}{\sum_{r=1}^M \sum_{l=1}^K \phi_r \pi_{r,l} \mathcal{N}(\mathcal{T}; \boldsymbol{\mu}_{r,l}^\mathcal{T}, \boldsymbol{\Sigma}_{r,l}^\mathcal{T})} \\ \hat{\boldsymbol{\mu}}_{o,m} &= \boldsymbol{\mu}_{o,m}^{\{s,\mathcal{F}\}} + \boldsymbol{\Sigma}_{o,m}^{\{s,\mathcal{F}\} \cdot \mathcal{T}} (\boldsymbol{\Sigma}_{o,m}^\mathcal{T})^{-1} (\mathcal{T} - \boldsymbol{\mu}_{o,m}^\mathcal{T}) \\ \hat{\boldsymbol{\Sigma}}_{o,m} &= \boldsymbol{\Sigma}_{o,m}^{\{s,\mathcal{F}\}} - \boldsymbol{\Sigma}_{o,m}^{\{s,\mathcal{F}\} \cdot \mathcal{T}} (\boldsymbol{\Sigma}_{o,m}^\mathcal{T})^{-1} \boldsymbol{\Sigma}_{o,m}^{\mathcal{T} \cdot \{s,\mathcal{F}\}} \end{aligned}$$

where the terms s, \mathcal{F} and \mathcal{T} in superscript denote the dimensions corresponding to s, \mathcal{F} and \mathcal{T} respectively.

Table 1: Computational complexity during motion generation with respect to the variables of interest.

The number of DMPs	$\mathcal{O}(n)$
The number of demonstrations	$\mathcal{O}(n)$
The number of GMM components	$\mathcal{O}(n)$
The number of task parameters	$\mathcal{O}(n^3)$
For constant task parameters	$\mathcal{O}(1)$

5 Related Work

In general there are two main approaches for learning motor skills; firstly those based on mimicking motion data using dynamical systems, i.e. DMPs [19, 26], secondly those relying on statistical machine learning, i.e. Gaussian Mixture Models (GMMs) [7] and Hidden Markov Models (HMMs) [16, 17]. DMPs consider one-shot learning and provide spatial and temporal scalability properties as well as guaranteed convergence to the goal position. Learning is done at acceleration level and many variations exist for DMPs [11, 20, 21]. Statistical machine learning based approaches directly learn on spatial data and can easily encode multiple demonstrations at a time, but lack certain properties presented by a DMP, for instance spatial amplification of the movement or guaranteed convergence to a goal position.

Among the GMM based approaches, [7] used different frames of reference for capturing distinct aspects of multiple demonstrations. Task parameters are defined as frames of reference. For generalization, the already learned GMMs are placed with respect to new frames and multiplied to retrieve a GMM for a new scenario. A similar method is [6], where they have shown that such an approach can also provide some extrapolation capability. However, these approaches lack typical properties associated with DMPs, e.g. spatial amplification of the movement and guaranteed convergence to a goal position. Additionally the frames of references cannot be arbitrarily placed and the user needs to have a certain level of understanding about how to place the frame of references for the TP-GMM to be successful. Probabilistic movement primitives (ProMPs) have been

shown to have better inference capabilities than a DMP and can combine or blend multiple demonstrations to achieve task specific generalizations. However, ProMPs require a large number of demonstrations for learning. According to [25], ProMPs have a high-dimensional covariance matrix and many free parameters. That is why they suffer from overfitting without a large number of demonstrations.

Task specific variations of DMPs are considered in [10, 18, 27, 28]. Basis targets can be extracted for the corresponding style parameters of each demonstration [18]. The basis targets and style parameters are combined to get appropriate basis functions. For generalization to a new situation with different task parameters, for example different height of an obstacle in a point to point reaching task, one has to provide appropriate style parameters. To get the style parameters for novel situations, the mapping from task parameters to style parameters is learned by supervised learning.

A more direct 2-Step (2S) approach is considered in [10, 28]. In the first step a mapping from task parameters to the DMP parameters is learned. It can either be learned by Locally Weighted Regression (**LWR**^{2S}), by placing local kernels along the task parameters [28], or a function approximator such as Gaussian Process Regression (**GPR**^{2S}) [10]. In the second step, the inferred DMP parameters are used for motion generation. An extension of [28] is [27], where they used Locally Weighted Projection Regression (LWPR) [30] for avoiding manual placement of the basis functions, which significantly reduces the required number of basis functions as compared with LWR. They also emphasized that any suitable function approximator can be used for directly encoding forcing terms of the DMPs, eliminating need to follow the 2-Step procedure. When using GPR for learning the direct encoding, they called their approach **GPR**^{1S}, where the superscript 1S and 2S emphasize on the 1-Step and 2-Step approaches respectively. These approaches can provide generalization capability when interpolating along the task parameters, but they do not perform well for very few demonstrations and cannot extrapolate beyond the demonstrated regions of task parameters, as we also show in our experiments.

In our work, the use of a generative model with a DMP may seem like a strange choice at first, as existing approaches use discriminative models [10, 18, 27, 28]. The discriminative models have been shown to yield lower asymptotic errors compared to their generative counterparts, but they also require higher amount of training data to reach those levels [14]. More specifically, a discriminative model requires $\mathcal{O}(n)$ training examples for reaching its asymptotic error while a gener-

ative model require $\mathcal{O}(\log n)$ [14]. This implies that, for few training examples, the generative model might have already reached its lowest asymptotic error, and thus performing better than a discriminative model. Since PbD focuses on learning from as few examples as possible, a generative model is indeed a useful choice.

6 Results

For all experiments the temperature schedule (β) which we use for annealing (DAEM) is [0.1 0.2 ... 1.0]. The regularization term ϵ for eigenvalues of the GMM covariances is set to 10^{-6} . For initializing GMMs the trajectories are equally segmented in time domain and then the Gaussians are calculated from the samples of each segment. The parameters of all models are empirically set.

6.1 Simulation of variable height obstacle avoidance

Our first experiment consists of a planar point to point reaching task with the variable height of the obstacles. If there is an obstacle in the way, the trajectory has to change according to the height of the obstacle. In this experiment the task parameter is defined as *the maximum height to avoid the obstacle*.¹ The goal of learning is to adapt a motion trajectory *for a new desirable height*. The demonstrations can be visualized in Figure 6a. There are only two demonstrations with 200 samples in each demonstration. The task parameters associated with these demonstrations are [0.0903, 0.1598]. Two DMPs are learned for generating motion in the x and y axis. We set the number of components in each GMM to 6. The two GMMs ($\epsilon = 10^{-6}$) corresponding to the forcing terms of DMPs for motion on the y-axis are shown in Figure 6b. Now we apply our prescribed approach with the components in the IDGMM empirically set to 15. This transforms the complete data GMMs as shown in Figure 6c. A single computer with Ubuntu 14.04, Intel Core i7 4790K QuadCore 4.0GHz and 16GiB memory took approximately 13s for fitting the the GMM in Matlab R2015b. The forcing terms of all DMPs can be predicted in less than 1 ms during the reproduction in simulation.

Comparison: The proposed approach is compared with **LWR**^{2S} [28], **GPR**^{1S} [27], **GPR**^{2S} [10] and **TP-GMM** [5]. Since we have used Gaussian Mixture Regression for direct prediction of forcing terms, we refer

¹ This chosen task parameter can be directly used for evaluation for the task performance. But the alternative choice of task parameter is also possible (such as the height of the obstacle) without any algorithmic changes.

Table 2: Task errors (simulation) with different approaches.

	\mathbf{GMR}^{1S}	\mathbf{GMR}^{1S} (without DAEM)	\mathbf{LWR}^{2S}	\mathbf{GPR}^{1S}	\mathbf{GPR}^{2S}	$\mathbf{TP-GMM}$
Interpolation						
ME (m)	0.0027	0.0067	0.0072	0.0015	0.0074	0.0055
SD (m)	0.0012	0.0046	0.0045	0.0010	0.0052	0.0028
Extrapolation						
ME (m)	0.0113	0.0539	0.0541	0.0407	0.0835	0.0159
SD (m)	0.0033	0.0272	0.0313	0.0265	0.0385	0.0034

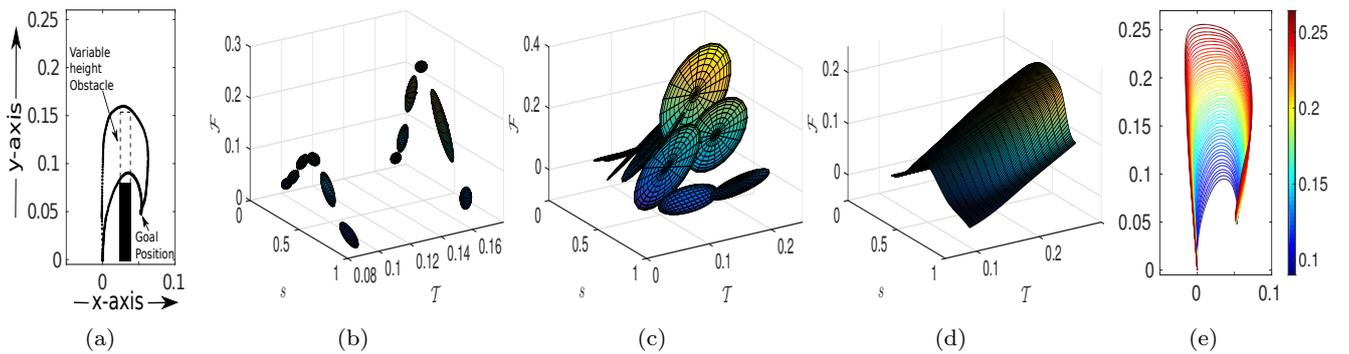


Fig. 6: A step by step illustration of TP-DMP learning with additional incomplete data and DAEM. (a) Demonstrations, (b) Initial GMMs encoding forcing terms of DMPs for y-axis, (c) transformed GMMs using incomplete data and DAEM, (d) learned regression surfaces for y-axis, (e) multiple generated movements.

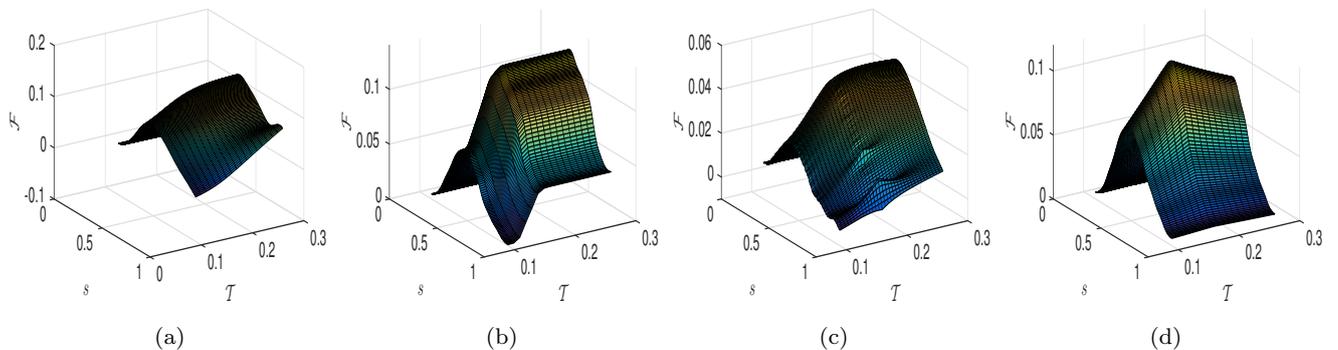


Fig. 7: Learned regression surfaces for y-axis (a) with \mathbf{GMR}^{1S} without DAEM, (b) with \mathbf{LWR}^{2S} , (c) with \mathbf{GPR}^{1S} and (d) with \mathbf{GPR}^{2S} .

to our approach as \mathbf{GMR}^{1S} . For \mathbf{LWR}^{2S} , [28] and [27] used tricubic kernel and Gaussian kernel respectively. We use Gaussian kernel with 5 equally spaced kernels placed along the task space. The choice of kernel is seldom important for the performance of LWR [28]. In \mathbf{LWR}^{2S} and \mathbf{GPR}^{2S} , the number of basis functions for shape parameters are set to 10, as the smaller values do not correctly capture the demonstrations. On the other hand, the GMMs in our approach require fewer num-

ber of Gaussians (i.e. 6) in this experiment. As in [27], GPR is used with the covariance function of the Matérn form, with isotropic distance measure and hyperparameters optimization [24]. Three frames of reference are defined for $\mathbf{TP-GMM}$, one at the starting point, one above the starting point with height equal to the desired height and one at the ending point. The number of components in the $\mathbf{TP-GMM}$ is empirically set to 4, as the higher values yield spiky motions. When per-

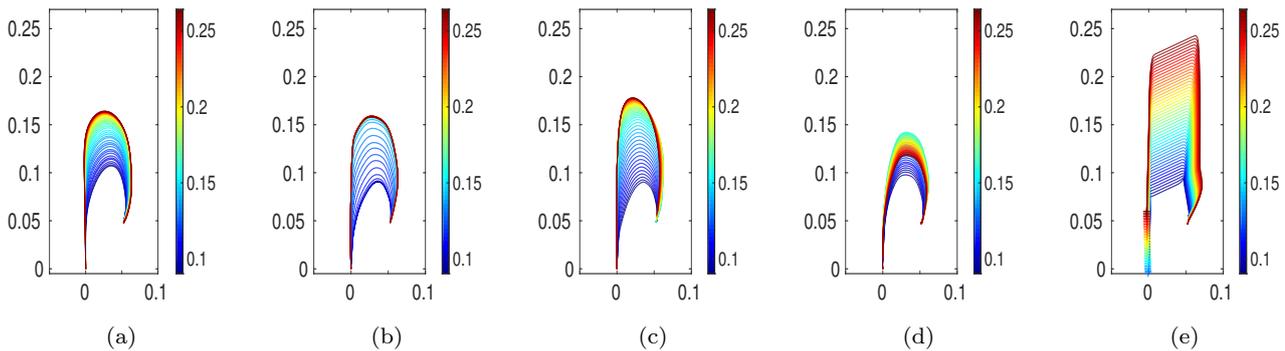


Fig. 8: Motion reproductions (a) with \mathbf{GMR}^{1S} without DAEM, (b) with \mathbf{LWR}^{2S} , (c) with \mathbf{GPR}^{1S} , (d) with \mathbf{GPR}^{2S} and (e) with $\mathbf{TP-GMM}$.

Table 3: Task errors (simulation) of different approaches when the height is changed during the trajectories.

	\mathbf{GMR}^{1S}	\mathbf{LWR}^{2S}	\mathbf{GPR}^{1S}	\mathbf{GPR}^{2S}	$\mathbf{TP-GMM}$
ME (m)	0.0038	0.0329	0.0239	0.0532	0.0129
SD (m)	0.0029	0.0253	0.0221	0.0342	0.0035

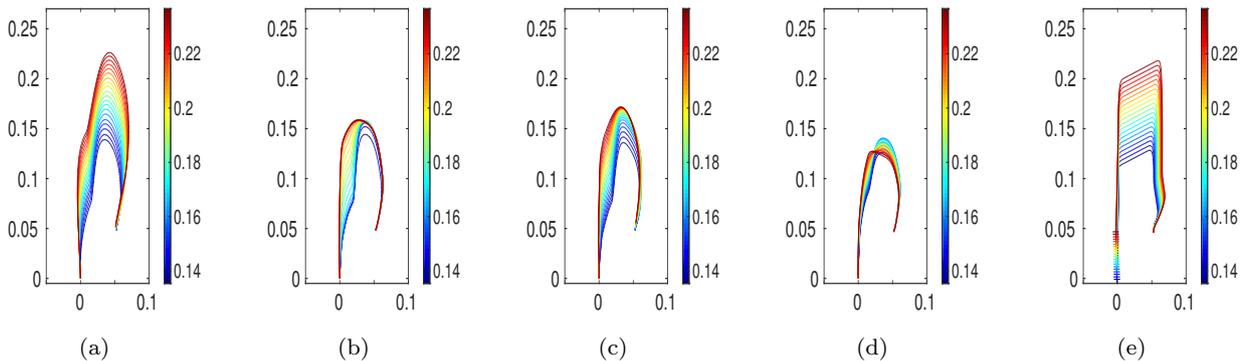


Fig. 9: Motion reproductions for changing the height during execution (a) with \mathbf{GMR}^{1S} , (b) with \mathbf{LWR}^{2S} , (c) with \mathbf{GPR}^{1S} , (d) with \mathbf{GPR}^{2S} and (e) with $\mathbf{TP-GMM}$.

forming learning with few demonstrations, if the GMMs in TP-GMM are learned with the high number of Gaussians then they converge to the data-points of the individual trajectories. Calculating the product of GMMs with Gaussians concentrated at regions of individual trajectories results in sudden activation of the responsibility term from one Gaussian to another Gaussian of a different trajectory and thus results the spiky behavior and wrong motion reproduction. Also the GMMs in TP-GMM is formed by encoding the relationship inbetween the clock signal and the spatial data with GMMs in different frame of references and their products afterwards while the GMMs in a TP-DMP model encode the relationship inbetween the clock signal, task parameters and the forcing term of a DMP. Since both models operate differently and encode different set of variables,

setting the same number of Gaussians in each is not needed for fair comparison.

The errors can be defined as the difference inbetween the maximum desired height of the trajectory (i.e. the input task parameter value used for the regression) and the actual achieved height values by the reproduced trajectories. We generate 50 linearly spaced task parameter values in the range $[\min(\mathcal{T}), \min(\mathcal{T}) + 2.5 \times (\max(\mathcal{T}) - \min(\mathcal{T}))]$ $([0.0903, 0.2641])$. The generated trajectories for these task parameters are shown in Figure 6e and 8. Table 2 presents the Mean absolute Error (ME) and its Standard Deviation (SD) when interpolating (task parameter in the range $[0.0903, 0.1577]$) and extrapolating (task parameter in the range $[0.1612, 0.2641]$). All approaches produce small errors when interpolating while \mathbf{GMR}^{1S} outperformed all other approaches when extrapolating beyond the demonstrated task param-

ters. The use of DAEM is critical for the performance of **GMR**^{1S} as the performance degrades substantially without annealing. The **TP-GMM** fails to preserve the shape information of the demonstrations, as it can be seen in Figure 8e.

Figure 6d and Figure 7 present the surface plots of the generated forcing terms ($\{s \times \mathcal{T}\} \mapsto \mathcal{F}$) of the DMP based approaches for y-axis. The regression surfaces of all the approaches except **GMR**^{1S} change their response in the extrapolation range, which is also the reason for their large errors when extrapolating. The regression surface of **GMR**^{1S} without DAEM shows that the EM converged to a poor local optima without annealing. The kernels in **LWR** and **GP** based approaches predict by mostly using the nearby data. Thus, their performance degrades when trying to extrapolate. The emphasis in **TP-GMM** model is to strictly pass through certain frames of reference and thus it can lose the shape information. Additionally, with the clock signal as the only input, the starting point of the reproductions with the **TP-GMM**, which is marked by a '—' sign in Figure 8e, moves quite far away from the starting point of the demonstrations. This problem can happen when the clock signal is the only input in the **TP-GMM** without consideration of the current point of a trajectory. A remedy to such a problem can be to encode the current point as an input.

Since we use a generative model for encoding the forcing terms of the DMP, our approach can benefit from the incomplete data. The IDGMM spans beyond the demonstrated range and thus retrieves a better underlying function when compared with the supervised learning approaches, which only rely on training data.

Varying task parameter: The task parameter is not necessary to be fixed during the reproduction phase and can vary if needed. Now 20 trajectories are generated with the initial desired task parameters linearly sampled from the interval $[0.0903, 0.1577]$. After one third of the executed motion, the desired task parameters are multiplied with 1.5. They now lie in the interval

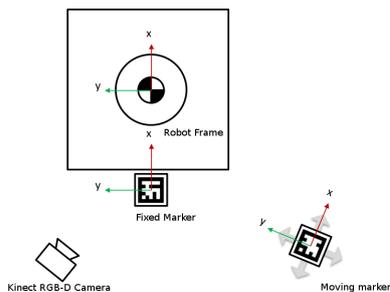


Fig. 10: Schematic of our vision system.

$[0.1355, 0.2366]$. Figure 9 contains the plot of the generated trajectories. A benefit of using a DMP-based approach is that the output trajectories are always smooth, even though the change in the task parameters is discontinuous. The reproduction errors are given in Table 3. Again, due to the aforementioned reasons, the **TP-DMP** outperforms all other approaches by producing least amount of error. Care should be taken in a real robot experiment, as an instantaneous change in the value of desired task parameters can cause a high acceleration at the end-effector. The high accelerations can be avoided by smoothly changing the task parameters when required.

6.2 Real robot experiments

Experimental setup: The experiments are conducted using a KUKA lightweight robot IV+. For collecting demonstrations, the robot is set to gravity compensation mode. With **GPR**^{1S}, offline trajectories are generated due to its high computational cost. This also means that with **GPR**^{1S}, the task parameters should remain fixed during the motion reproduction. The markers are tracked with Kinect RGB-D camera by using ROS wrapper for Alvar, an open source augmented reality tag tracking library [1]. More specifically, we have a fixed marker with respect to robot's frame of reference and a moving marker. The fixed marker is used for localizing the camera while the moving marker is used for tracking objects, as shown in Figure 10. A low pass filter is applied to remove high frequency noise in the vision system. The GMM model is always learned offline in Matlab. A single computer with Ubuntu 12.04 32-bit, Intel Core i5-2500 quad core 3.3GHz and 16GiB memory is used for marker tracking as well as online motion generation. For motion reproduction with **GMR**^{1S}, the forcing terms of all DMPs are predicted within 7 ms. A Cartesian impedance controller with a control frequency of 100Hz is used for motion generation.

6.2.1 Sweeping task:

This experiment considers a sweeping task, which consists of moving trash to a collection point. The task parameters are defined as *the planar coordinates of the trash*. A teacher physically holds the end-effector for various trash positions and demonstrates the required motions for moving the trash to the collection point, as shown in Figure 11a. Learning is performed in task space (position and orientation of the end-effector). Each demonstration has a duration of approximately 5 seconds with a sampling rate of 10ms.

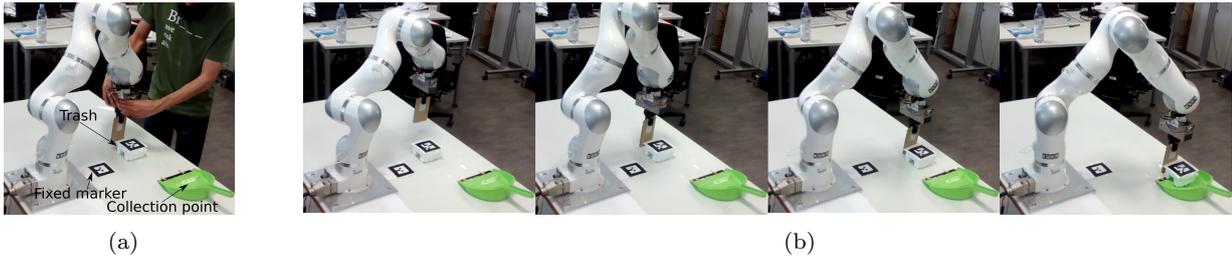


Fig. 11: Demonstrations are collected by setting the robot in gravity compensation mode while a Cartesian impedance controller is used for motion generation. (a) A human provides the demonstration for moving the trash to the dustpan. (b) The robot generates motion for a new trash position.

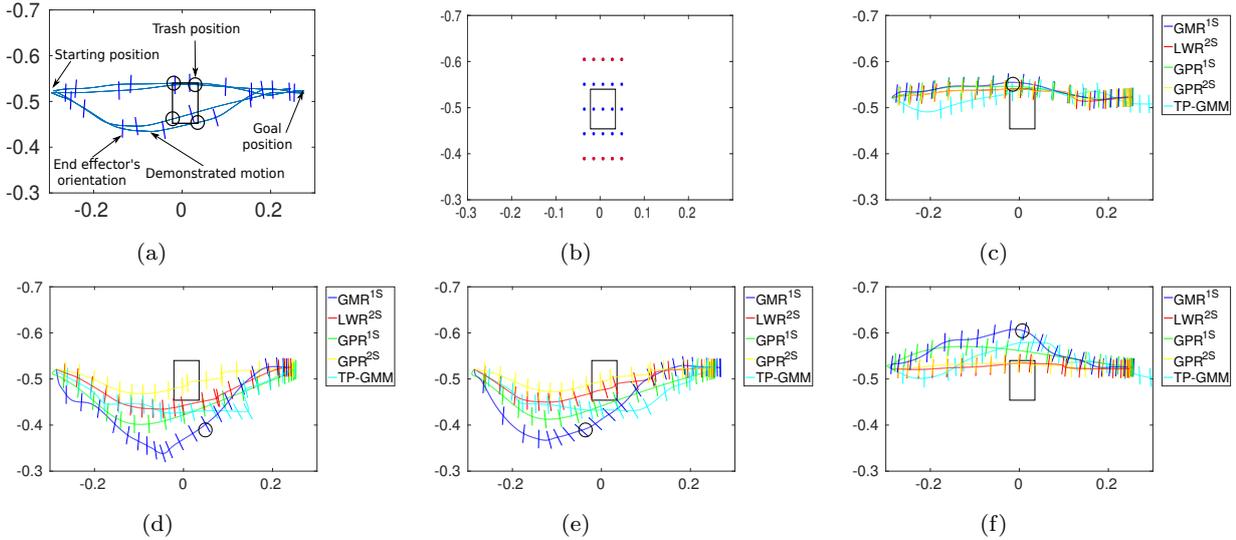


Fig. 12: Comparison of \mathbf{GMR}^{1S} , \mathbf{LWR}^{2S} , \mathbf{GPR}^{1S} , \mathbf{GPR}^{2S} and $\mathbf{TP-GMM}$ for the sweeping task. (a) Demonstrations for the sweeping task. Circles represent the trash positions while the rectangle represents the bounding box enclosing these trash positions. (b) Blue dots represent trash positions for interpolation evaluation while the red ones are for extrapolation evaluations. (c,d,e,f) Generated movements for new trash positions.

Three DMPs are learned, two for generating a planar motion and one for encoding the planar orientation of the end-effector. As shown in Figure 12a, four demonstrations are provided for different positions of the trash. In our demonstrations, the x and y values of the trash position lies between $[-0.0216m, 0.0350m]$ and $[-0.54m, -0.454m]$ respectively (drawn as a rectangle). We selected 25 new trash positions (a grid) for evaluation in the extended x and y ranges of $[-0.0358m, 0.0491m]$ and $[-0.6045m, -0.3895m]$ respectively, which can be visualized in Figure 12b. The blue samples, which are close to the bounding box of the demonstrated region, are used for evaluating interpolation performance. The red samples, which are far away from the bounding box, are used for evaluating extrapolation performance.

Comparison: We defined error as the minimum distance inbetween the trash position and the generated trajectory. Table 4 contains the ME and its SD when us-

ing our approach, \mathbf{LWR}^{2S} , \mathbf{GPR}^{1S} , \mathbf{GPR}^{2S} and $\mathbf{TP-GMM}$. Three frames at starting point, ending point and at the trash location are defined for $\mathbf{TP-GMM}$. The number of components in the $\mathbf{TP-GMM}$ is empirically set to 6, as the higher values yield spiky motion. The basis functions in \mathbf{LWR}^{2S} and \mathbf{GPR}^{2S} are set to 60. For our approach the components in each GMM, as well as the IDGMM, are set to 40. The remaining settings are same as in previous experiment. Like in the previous experiment, our approach requires less components as compared with the basis functions in the other approaches. Again, our approach produces less errors for interpolation as well as extrapolation as shown in Figures (12c-12f) and Table 4. The error for other approaches increases considerably as we move away from the demonstrated interval. Additionally, some of the trajectories generated by $\mathbf{TP-GMM}$ surpassed the collection points as the $\mathbf{TP-GMM}$ model does not have the notion of a goal position. Interestingly, no demon-

Table 4: Task errors (KUKA) with different approaches.

	GMR ^{1S}	LWR ^{2S}	GPR ^{1S}	GPR ^{2S}	TP-GMM
Interpolation					
ME (m)	5.4×10^{-3}	13×10^{-3}	7.5×10^{-3}	14.5×10^{-3}	11×10^{-3}
SD (m)	4.3×10^{-3}	7.1×10^{-3}	4.3×10^{-3}	10.3×10^{-3}	5.3×10^{-3}
Extrapolation					
ME (m)	9.9×10^{-3}	68.3×10^{-3}	43×10^{-3}	84.6×10^{-3}	33.8×10^{-3}
SD (m)	9.6×10^{-3}	6.3×10^{-3}	5.7×10^{-3}	13×10^{-3}	8.2×10^{-3}

strations are provided for the trash positions in the upper half of the sweeping area, which makes it an interesting region for comparing different approaches. **GMR**^{1S} also successfully generates the motion for the trash position in this region, as shown in Figure 12f.

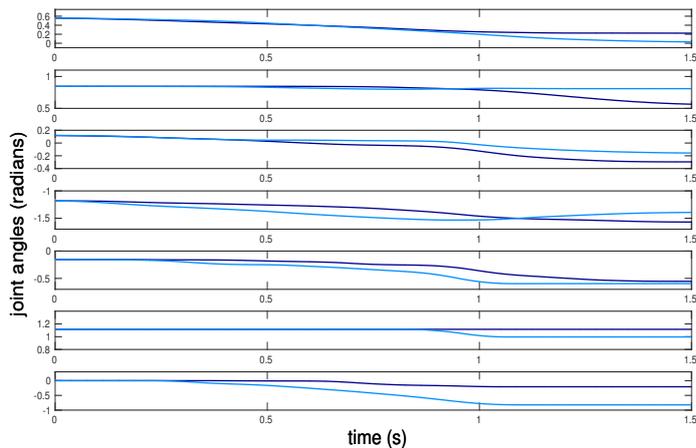
6.2.2 Striking task:

This experiment considers a task which involves striking a ball such that it hits a desired target position. The ball is always placed at the same point and the task parameter is defined as the *the x-coordinate of the target*. The y-coordinate of the target is fixed in the two demonstrations. Similar to the previous experiment, a teacher physically holds the end-effector and demonstrates the required motion for hitting the target. The demonstrations have a duration of appropriately 1.5 seconds with a sampling rate of 10ms. Due to the small duration of the motions we increase the size of data ten times (to approximately 1500) by inserting samples in between the adjacent data points of the trajectories by using linear interpolation. Now we consider learning in joint space. Different demonstrations can produce completely different final joints configuration during the demonstrations. The final joint configurations are measurable in the demonstrations but are unknown when reproducing motion for a new target position. Thus, it is necessary to predict the final joints configuration during the reproduction phase. Seven DMPs are learned with one DMP for each joint of the robot and thus utilizing all DOFs.

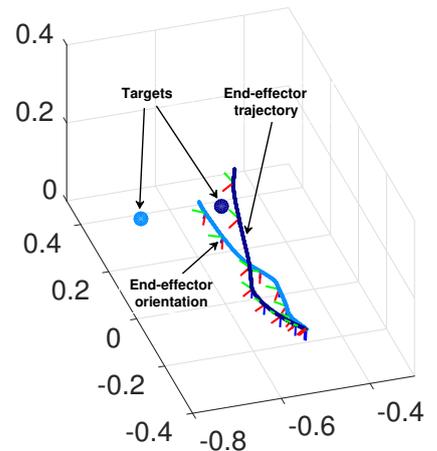
Our approach can also easily incorporate the learning of meta parameters in a DMP. For **GMR**^{1S}, we learn the distribution of $(s, [\mathcal{T} g], \mathcal{F})$. Similar to task parameters, the goal terms g (meta parameters) are constants for a single demonstration. Thus we simply interpret them as additional task parameters. This also means that different DOF in each demonstration will now have different task parameters. The final joint configurations (goal positions), which we set as an addi-

tional task parameter, cannot be known in advance. As there is no distinction inbetween input and output variables when fitting a GMM and during GMR, any set of variables can be selected as input, to retrieve the expected value of remaining variables. Thus with GMR, the observable task variable can be used for predicting the expected value of missing task variables and for motion generation. So now, with GMR, we not only predict the goal terms g of each DMP but also generate the forcing terms. For **GMR**^{1S} the components in each GMM ($\varepsilon = 10^{-4}$), as well as in the IDGMM, are set to 60. The number of basis functions in **LWR**^{2S} and **GPR**^{2S} is set to 100. The goal positions for **LWR**^{2S} and **GPR**^{2S} are predicted in the first step along with the DMP parameters by using LWR and GPR respectively. The goal position in **GPR**^{1S} is predicted at each time step along with the forcing terms by using GPR. Two frames of reference are defined for **TP-GMM**: one at the start of the trajectory and one at the end of the trajectory. As mentioned before, the final joint configurations are not known and hence the **TP-GMM** cannot be used directly in this experiment. To use **TP-GMM**, we first predict the final joint configuration with GP and then the second frame of reference is placed at that final joint configuration for motion reproduction i.e. the offset vector for second frame of reference looks like $b_2 = [t_f j_{f1} j_{f2} j_{f3} j_{f4} j_{f5} j_{f6} j_{f7}]'$ where t_f is the final time value and j_{fn} is the predicted final joint angle for the n^{th} joint. The transformation matrices for the two frames of references are set to identity matrices. The number of components in the **TP-GMM** is empirically set to 4. The remaining settings are same as in the previous experiments.

Comparison: A binary evaluation criterion is defined as a success if the robot is able to hit the target and failure otherwise. Demonstrations are provided for hitting a target with x-coordinate of $-0.4891m$ and $-0.6703m$, as shown in Figure 13a. Figure 13b shows the end-effector pose. Interpolation performance is evaluated for the target x-coordinates of $-0.5663m$ while

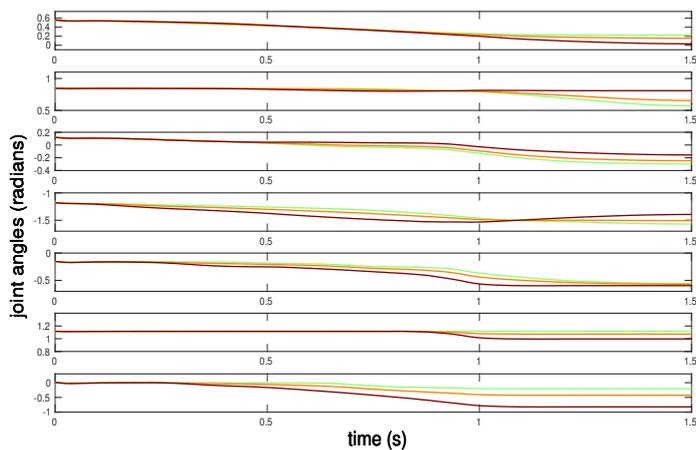


(a) Demonstrated joint angles in radians and time in seconds.

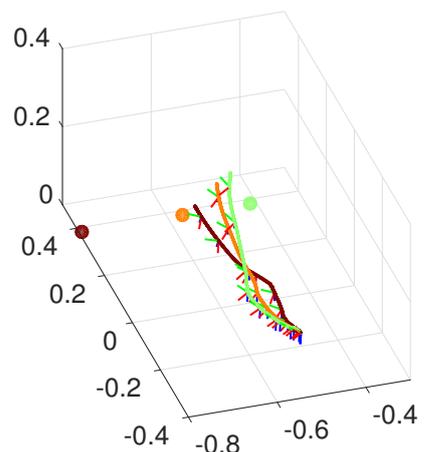


(b) End-effector trajectory in Cartesian space.

Fig. 13: Joint angles and end-effector trajectory of the demonstrated motions.



(a) Reproduced joint angles.



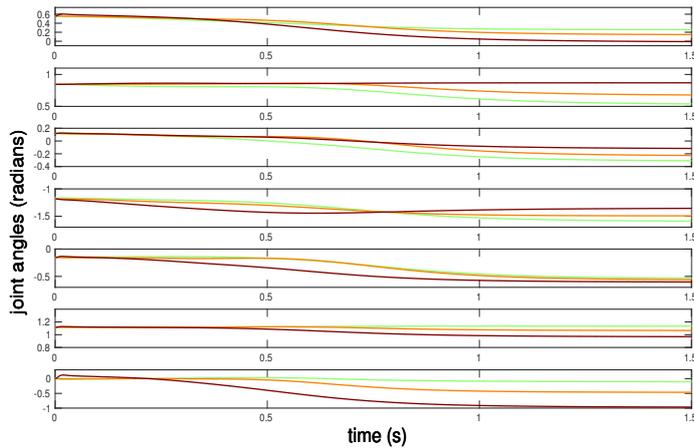
(b) End-effector trajectory in Cartesian space.

Fig. 14: Reproductions with \mathbf{LWR}^{2S}

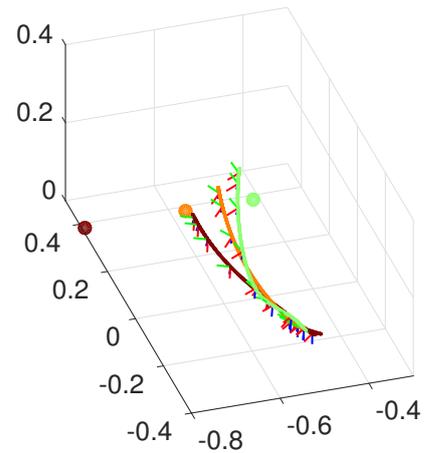
extrapolation performance is evaluated for the target x-coordinates of $-0.4146m$ and $-0.7933m$. When generalizing for novel goal positions, the DMPs can produce high accelerations at the beginning of the movement [15]. This is due to the initial interaction between the linear dynamics and forcing terms. This undesirable behavior was slightly observed in this experiment. A simple solution to solve this problem is to gradually activate the forcing terms. Thus we multiply the predicted forcing terms by $(1 - s^{10})$. As s decays exponentially, the effect of this term vanishes very quickly.

Figure 14 contains the reproduction results with \mathbf{LWR}^{2S} . It produces a good trajectory for interpolation performance as it lies in between the demonstrated trajectories. The trajectories for the extrapolation fails to

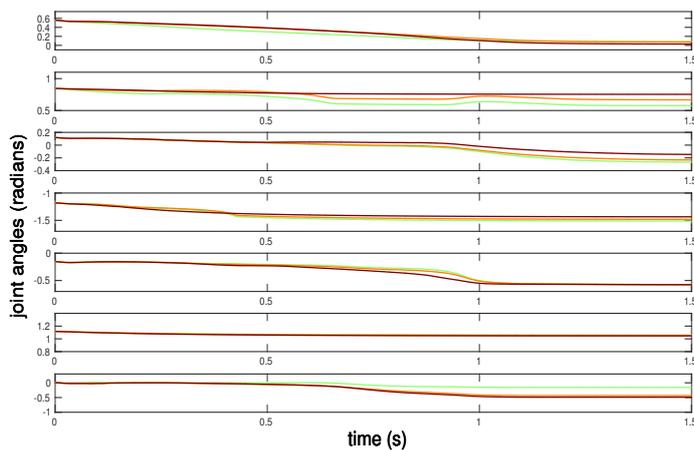
reproduce the task as they are similar to the demonstrations. The \mathbf{LWR}^{2S} also encountered the same problem during the first experiment where successful trajectories were generated for interpolation intervals but it failed to reproduce during the extrapolation intervals. Figure 15 contains the reproduction results with \mathbf{GPR}^{1S} . It can easily be observed that the initial end-effector trajectory required to approach the ball (and critical for the execution of the task) is very similar for the three reproductions. The trajectories lose the important shape information required for the successful execution of the task. Also the green and brown trajectories of one of the joints are almost overlapping. Since the reproduction is in joint space, an incorrect motion of even a single joint can lead to the failure of the task. Figure 16



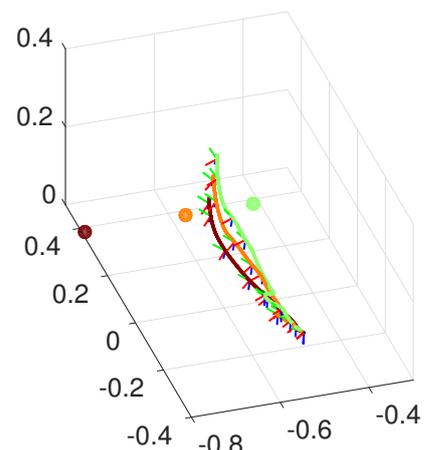
(a) Reproduced joint angles.



(b) End-effector trajectory in Cartesian space.

Fig. 15: Reproductions with \mathbf{GPR}^{1S} 

(a) Reproduced joint angles.



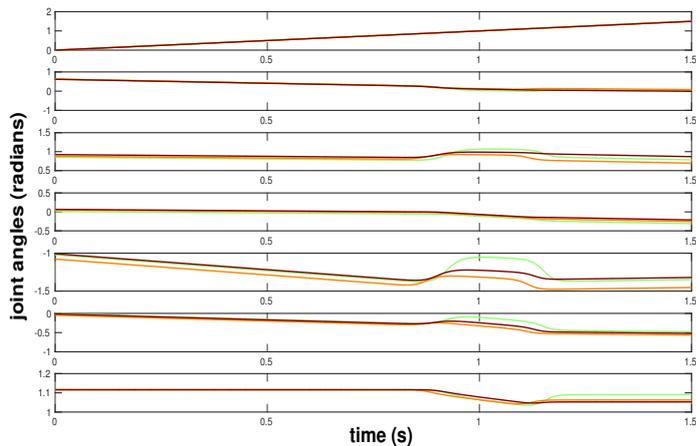
(b) End-effector trajectory in Cartesian space.

Fig. 16: Reproductions with \mathbf{GPR}^{2S}

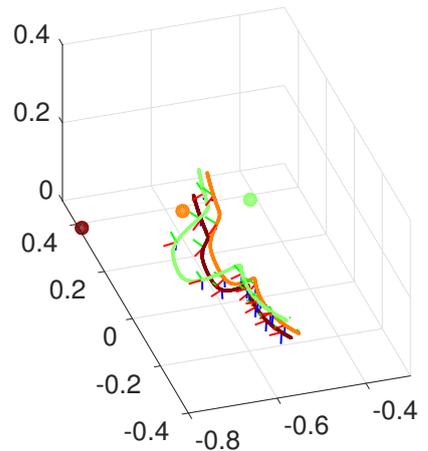
contains the reproduction results with \mathbf{GPR}^{2S} . Some of the reproduced trajectories are very different from the demonstrated ones. As with \mathbf{GPR}^{1S} , the trajectories generated with \mathbf{GPR}^{2S} lose the initial shape information which is required for the successful execution of the task. The failure of both of the GP based approaches can be attributed the small amount of training data, i.e. only two trajectories. Figure 17 contains the reproduction results with $\mathbf{TP-GMM}$. The reproduced trajectories do not capture the demonstrated motions and it fails to learn anything useful for this task. The joint distribution of all the eight variables (one phase and seven joint angles) is encoded in the $\mathbf{TP-GMM}$. As we only have two trajectories, the product of GMMs in

$\mathbf{TP-GMM}$ suffers from a severe curse of dimensionality.

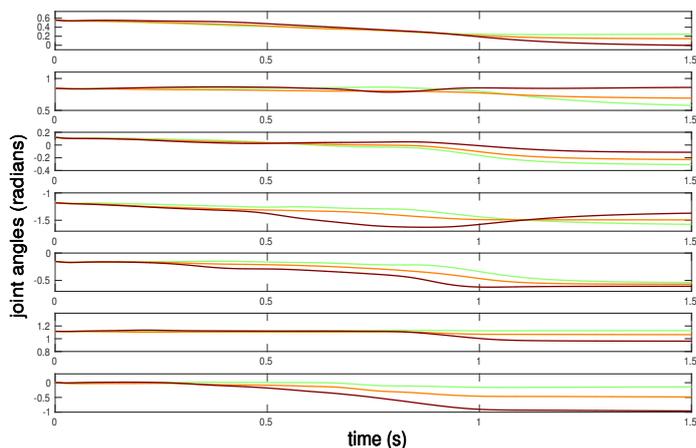
Figure 18 contains the reproduction results with \mathbf{GMR}^{1S} . The shape information is preserved in the reproduced trajectories and the DMPs goal parameters are correctly inferred. The generated joint angles trajectories extend further away from the demonstrated trajectories for extrapolation. The joint angles trajectories for interpolation are inbetween the extrapolation trajectories. Executing motion trajectories generated by \mathbf{GMR}^{1S} on KUKA show that our approach always yields success in the extended range of [0.4146, 0.7933]. For the two extrapolation evaluations with our approach, the ball trajectories, as well as the executed motions on KUKA, are visualized in Figure 19, where



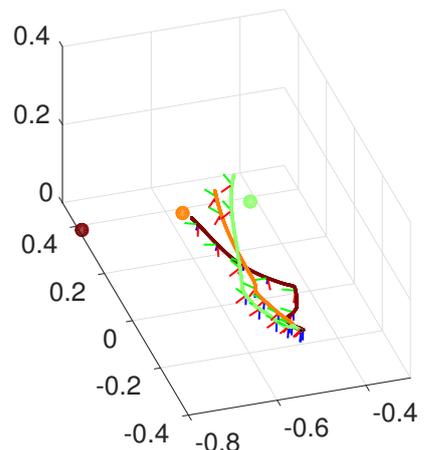
(a) Reproduced joint angles.



(b) End-effector trajectory in Cartesian space.

Fig. 17: Reproductions with **TP-GMM**

(a) Reproduced joint angles.



(b) End-effector trajectory in Cartesian space.

Fig. 18: Reproductions with **GMR**^{1S}

the different final joint configurations of the reproduced motions are also observed.

7 Conclusion and future work

We have shown how the task specific generalization of a DMP can be achieved by formulating learning as a density estimation problem. The proposed approach captures the local behavior of each demonstration by using a GMM. These GMMs are then mixed to get the task specific generalization. We have handled the data sparsity along task parameters by introducing additional incomplete data filling the input space. Deterministic Annealing EM is used to avoid the local maxima problem.

We retain the local behavior of each GMM by keeping the means and mixing weights within the GMMs fixed. The task specific generalization is achieved by just adapting covariances and mixing coefficient of the already learned GMMs. The TP-DMP framework can perform learning in task space as well as in joint space and can even handle the learning of meta parameters of a DMP. As shown in the experiments, our approach requires very few demonstrations for learning and it outperforms the existing approaches specially when extrapolating beyond the demonstrated ranges of the task variables. As future work, we plan to extend our proposed work with sample reuse approach [8] and to investigate the scalability issue to complex tasks.

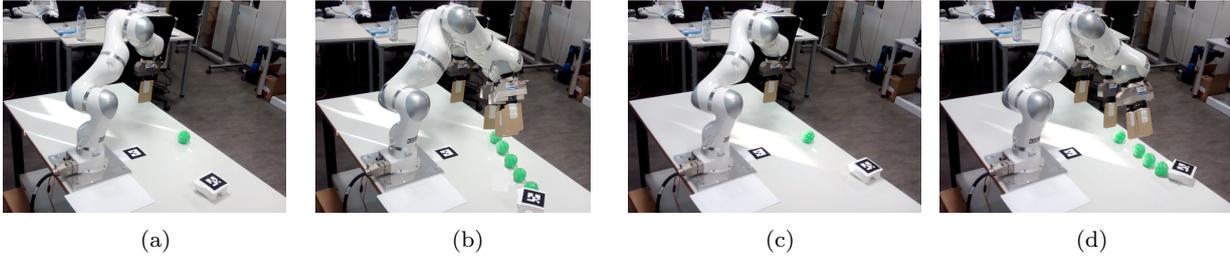


Fig. 19: Executed motions for the two extrapolation evaluations. (a,c) Robot initial configuration and different target positions. (b) Executed striking motion for hitting the target in (a). (d) Executed striking motion for hitting the target in (c).

Appendix

Following properties have been used in this proof [23]:

- *Property 1:* If $\mathbf{X} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ and \mathbf{b} a vector then $E[(\mathbf{X} + \mathbf{b})(\mathbf{X} + \mathbf{b})^\top] = \boldsymbol{\Sigma} + (\boldsymbol{\mu} + \mathbf{b})(\boldsymbol{\mu} + \mathbf{b})^\top$
- *Property 2:* The product of Gaussian densities is defined as: $\mathcal{N}(\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1)\mathcal{N}(\boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2) = e \mathcal{N}(\boldsymbol{\mu}_e, \boldsymbol{\Sigma}_e)$ where $e = \mathcal{N}(\boldsymbol{\mu}_1; \boldsymbol{\mu}_2, \boldsymbol{\Sigma}_1 + \boldsymbol{\Sigma}_2)$.

Expected pdf value:

The Expected pdf value of a Gaussian $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ evaluated at a Gaussian stochastic random variable $\mathbf{X} \sim \mathcal{N}(\boldsymbol{\mu}_X, \boldsymbol{\Sigma}_X)$ can be defined as:

$E[\mathcal{N}(\mathbf{X}; \boldsymbol{\mu}, \boldsymbol{\Sigma})] \rightarrow E[\mathcal{N}(\mathcal{N}(\boldsymbol{\mu}_X, \boldsymbol{\Sigma}_X); \boldsymbol{\mu}, \boldsymbol{\Sigma})]$ which is similar to $E[f(\mathbf{X})] = \int f(x)p(x)dx$. Now

$$E[\mathcal{N}(\mathcal{N}(\boldsymbol{\mu}_X, \boldsymbol{\Sigma}_X); \boldsymbol{\mu}, \boldsymbol{\Sigma})] = \int_{-\infty}^{\infty} \dots \int_{-\infty}^{\infty} \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})\mathcal{N}(\boldsymbol{\mu}_X, \boldsymbol{\Sigma}_X)$$

By using property 2.

$$\begin{aligned} E[\mathcal{N}(\mathcal{N}(\boldsymbol{\mu}_X, \boldsymbol{\Sigma}_X); \boldsymbol{\mu}, \boldsymbol{\Sigma})] &= \\ &= \int_{-\infty}^{\infty} \dots \int_{-\infty}^{\infty} e\mathcal{N}(x; \boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c) \\ &= e \int_{-\infty}^{\infty} \dots \int_{-\infty}^{\infty} \mathcal{N}(x; \boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c) \\ &= e \end{aligned}$$

where

$$e = \mathcal{N}(\boldsymbol{\mu}; \boldsymbol{\mu}_X, \boldsymbol{\Sigma} + \boldsymbol{\Sigma}_X) \quad (3)$$

Incorporating Incomplete data via EM for mixture of GMMs:

In the E-step, the expectations $E[z_{i,k}|\mathbf{x}^{obs}, \boldsymbol{\theta}_t]$, $E[z_{i,k}, \mathbf{x}^{miss}|\mathbf{x}^{obs}, \boldsymbol{\theta}_t]$ and $E[z_{i,k}, \mathbf{x}^{miss}\mathbf{x}^{miss\top}|\mathbf{x}^{obs}, \boldsymbol{\theta}_t]$ have to be calculated for incomplete data [12]. Let $\boldsymbol{\mu}_{o,m} = \begin{bmatrix} \boldsymbol{\mu}_{o,m}^{obs} \\ \boldsymbol{\mu}_{o,m}^{miss} \end{bmatrix}$, $\boldsymbol{\Sigma}_{o,m} = \begin{bmatrix} \boldsymbol{\Sigma}_{o,m}^{obs} & \boldsymbol{\Sigma}_{o,m}^{obs,miss} \\ \boldsymbol{\Sigma}_{o,m}^{miss,obs} & \boldsymbol{\Sigma}_{o,m}^{miss} \end{bmatrix}$, $\mathbf{X}_w = \begin{bmatrix} \mathbf{X}_w^{obs} \\ \mathbf{X}_w^{miss} \end{bmatrix}$ where $\mathbf{X}_w^{obs} \sim \mathcal{N}(\boldsymbol{\mu}_{o,m}^{obs}, \boldsymbol{\Sigma}_{o,m}^{obs})$ is the observable part representing the GMM components spanning the input space. Now the similar expectations of the missing

dimensions (output) are calculated as:

$$\begin{aligned} E[z_{w,o,m}|\mathbf{X}_w^{obs}, \boldsymbol{\theta}_t] &= d_{w,o,m}^{t+1} \\ &= \frac{c_{w,o,m}^{t+1}}{\sum_{r=1}^M \sum_{l=1}^K c_{w,r,l}^{t+1}} \times \boldsymbol{\pi}_w^{obs} \times N \end{aligned}$$

with $c_{w,o,m}^{t+1} = (\phi_o \boldsymbol{\pi}_{o,m} \mathcal{N}(\boldsymbol{\mu}_w^{obs}; \boldsymbol{\mu}_{o,m}^{obs}, \boldsymbol{\Sigma}_w^{obs} + \boldsymbol{\Sigma}_{o,m}^{obs}))^\beta$ where $\mathcal{N}(\boldsymbol{\mu}_w^{obs}; \boldsymbol{\mu}_{o,m}^{obs}, \boldsymbol{\Sigma}_w^{obs} + \boldsymbol{\Sigma}_{o,m}^{obs})$ is the expected pdf calculated only on the observed dimensions, as derived for Equation (3).

$$E[z_{w,o,m}, \mathbf{X}_w^{miss}|\mathbf{X}_w^{obs}, \boldsymbol{\theta}_t] = d_{w,o,m}^{t+1} (\boldsymbol{\mu}_{o,m}^{miss} + \boldsymbol{\Sigma}_{o,m}^{miss,obs} (\boldsymbol{\Sigma}_{o,m}^{obs})^{-1} (\mathbf{X}_w^{obs} - \boldsymbol{\mu}_{o,m}^{obs}))$$

Define: $\hat{\mathbf{X}}_{w,o,m}^{miss} = \boldsymbol{\mu}_{o,m}^{miss} + \boldsymbol{\Sigma}_{o,m}^{miss,obs} (\boldsymbol{\Sigma}_{o,m}^{obs})^{-1} (\mathbf{X}_w^{obs} - \boldsymbol{\mu}_{o,m}^{obs})$

$$\begin{aligned} E[z_{w,o,m}, \mathbf{X}_w^{miss}|\mathbf{X}_w^{obs}, \boldsymbol{\theta}_t] &= d_{w,o,m}^{t+1} \hat{\mathbf{X}}_{w,o,m}^{miss} \\ E[z_{w,o,m}, \mathbf{X}_w^{miss} \mathbf{X}_w^{miss\top}|\mathbf{X}_w^{obs}, \boldsymbol{\theta}_t] &= \\ &= d_{w,o,m}^{t+1} (\boldsymbol{\Sigma}_{o,m}^{miss} - \boldsymbol{\Sigma}_{o,m}^{miss,obs} (\boldsymbol{\Sigma}_{o,m}^{obs})^{-1} (\boldsymbol{\Sigma}_{o,m}^{miss,obs})^\top + \\ &\quad \hat{\mathbf{X}}_{w,o,m}^{miss} \hat{\mathbf{X}}_{w,o,m}^{miss\top}) \end{aligned}$$

Now for updating $\boldsymbol{\Sigma}_{o,m}^{t+1}$, the term

$d_{w,o,m}^{t+1} (\mathbf{X}_w - \boldsymbol{\mu}_{o,m}) (\mathbf{X}_w - \boldsymbol{\mu}_{o,m})^\top$ for incomplete data is calculated as:

$$\begin{aligned} d_{w,o,m}^{t+1} \begin{bmatrix} \mathbf{X}_w^{obs} - \boldsymbol{\mu}_{o,m}^{obs} \\ \mathbf{X}_w^{miss} - \boldsymbol{\mu}_{o,m}^{miss} \end{bmatrix} \begin{bmatrix} \mathbf{X}_w^{obs} - \boldsymbol{\mu}_{o,m}^{obs} \\ \mathbf{X}_w^{miss} - \boldsymbol{\mu}_{o,m}^{miss} \end{bmatrix}^\top \\ = \begin{bmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} \\ \mathbf{A}_{21} & \mathbf{A}_{22} \end{bmatrix} \end{aligned}$$

where

$$\begin{aligned} \mathbf{A}_{11} &= d_{w,o,m}^{t+1} (\mathbf{X}_w^{obs} - \boldsymbol{\mu}_{o,m}^{obs}) (\mathbf{X}_w^{obs} - \boldsymbol{\mu}_{o,m}^{obs})^\top \\ &\quad \text{by using Property 1} \\ &= d_{w,o,m}^{t+1} (\boldsymbol{\Sigma}_w^{obs} + (\boldsymbol{\mu}_w^{obs} - \boldsymbol{\mu}_{o,m}^{obs})(\boldsymbol{\mu}_w^{obs} - \boldsymbol{\mu}_{o,m}^{obs})^\top) \end{aligned}$$

$$\begin{aligned}
\mathbf{A}_{21} &= d_{w,o,m}^{t+1} (\mathbf{X}_w^{miss} - \boldsymbol{\mu}_{o,m}^{miss}) (\mathbf{X}_w^{obs} - \boldsymbol{\mu}_{o,m}^{obs})^\top \\
&= (E[z_{w,o,m}, \mathbf{X}_w^{miss}] - d_{w,o,m}^{t+1} \boldsymbol{\mu}_{o,m}^{miss}) (\mathbf{X}_w^{obs} - \boldsymbol{\mu}_{o,m}^{obs})^\top \\
&= \left(d_{w,o,m}^{t+1} (\boldsymbol{\mu}_{o,m}^{miss} + \boldsymbol{\Sigma}_{o,m}^{miss.obs} (\boldsymbol{\Sigma}_{o,m}^{obs})^{-1} \right. \\
&\quad \left. (\mathbf{X}_w^{obs} - \boldsymbol{\mu}_{o,m}^{obs})) - d_{w,o,m}^{t+1} \boldsymbol{\mu}_{o,m}^{miss} \right) (\mathbf{X}_w^{obs} - \boldsymbol{\mu}_{o,m}^{obs})^\top \\
&= d_{w,o,m}^{t+1} \boldsymbol{\Sigma}_{o,m}^{miss.obs} (\boldsymbol{\Sigma}_{o,m}^{obs})^{-1} \\
&\quad (\mathbf{X}_w^{obs} - \boldsymbol{\mu}_{o,m}^{obs}) (\mathbf{X}_w^{obs} - \boldsymbol{\mu}_{o,m}^{obs})^\top \\
&= d_{w,o,m}^{t+1} \boldsymbol{\Sigma}_{o,m}^{miss.obs} (\boldsymbol{\Sigma}_{o,m}^{obs})^{-1} \\
&\quad \left(\boldsymbol{\Sigma}_w^{obs} + (\boldsymbol{\mu}_w^{obs} - \boldsymbol{\mu}_{o,m}^{obs}) (\boldsymbol{\mu}_w^{obs} - \boldsymbol{\mu}_{o,m}^{obs})^\top \right)
\end{aligned}$$

$$\mathbf{A}_{12} = \mathbf{A}_{21}^\top$$

$$\begin{aligned}
\mathbf{A}_{22} &= d_{w,o,m}^{t+1} (\mathbf{X}_w^{miss} - \boldsymbol{\mu}_{o,m}^{miss}) (\mathbf{X}_w^{miss} - \boldsymbol{\mu}_{o,m}^{miss})^\top \\
&= E[z_{w,o,m}, \mathbf{X}_w^{miss} \mathbf{X}_w^{miss}^\top] + d_{w,o,m}^{t+1} \boldsymbol{\mu}_{o,m}^{miss} (\boldsymbol{\mu}_{o,m}^{miss})^\top \\
&\quad - 2E[z_{w,o,m}, \mathbf{X}_w^{miss}] (\boldsymbol{\mu}_{o,m}^{miss})^\top \\
&= d_{w,o,m}^{t+1} (\boldsymbol{\Sigma}_{o,m}^{miss} - \boldsymbol{\Sigma}_{o,m}^{miss.obs} (\boldsymbol{\Sigma}_{o,m}^{obs})^{-1} (\boldsymbol{\Sigma}_{o,m}^{miss.obs})^\top \\
&\quad + \hat{\mathbf{X}}_{w,o,m}^{miss} \hat{\mathbf{X}}_{w,o,m}^{miss}^\top) + d_{w,o,m}^{t+1} \boldsymbol{\mu}_{o,m}^{miss} (\boldsymbol{\mu}_{o,m}^{miss})^\top \\
&\quad - 2 \left(d_{w,o,m}^{t+1} (\boldsymbol{\mu}_{o,m}^{miss} + \boldsymbol{\Sigma}_{o,m}^{miss.obs} (\boldsymbol{\Sigma}_{o,m}^{obs})^{-1} \right. \\
&\quad \left. (\mathbf{X}_w^{obs} - \boldsymbol{\mu}_{o,m}^{obs})) \right) (\boldsymbol{\mu}_{o,m}^{miss})^\top
\end{aligned}$$

$$\begin{aligned}
\mathbf{A}_{22} &= d_{w,o,m}^{t+1} \boldsymbol{\Sigma}_{o,m}^{miss} - d_{w,o,m}^{t+1} \boldsymbol{\Sigma}_{o,m}^{miss.obs} (\boldsymbol{\Sigma}_{o,m}^{obs})^{-1} (\boldsymbol{\Sigma}_{o,m}^{miss.obs})^\top \\
&\quad + d_{w,o,m}^{t+1} (\boldsymbol{\mu}_{o,m}^{miss} + \boldsymbol{\Sigma}_{o,m}^{miss.obs} (\boldsymbol{\Sigma}_{o,m}^{obs})^{-1} (\mathbf{X}_w^{obs} - \boldsymbol{\mu}_{o,m}^{obs})) \\
&\quad (\boldsymbol{\mu}_{o,m}^{miss} + \boldsymbol{\Sigma}_{o,m}^{miss.obs} (\boldsymbol{\Sigma}_{o,m}^{obs})^{-1} (\mathbf{X}_w^{obs} - \boldsymbol{\mu}_{o,m}^{obs}))^\top \\
&\quad + d_{w,o,m}^{t+1} \boldsymbol{\mu}_{o,m}^{miss} \boldsymbol{\mu}_{o,m}^{miss}^\top - 2d_{w,o,m}^{t+1} \boldsymbol{\mu}_{o,m}^{miss} \boldsymbol{\mu}_{o,m}^{miss}^\top \\
&\quad - 2d_{w,o,m}^{t+1} \boldsymbol{\Sigma}_{o,m}^{miss.obs} (\boldsymbol{\Sigma}_{o,m}^{obs})^{-1} (\mathbf{X}_w^{obs} - \boldsymbol{\mu}_{o,m}^{obs}) \boldsymbol{\mu}_{o,m}^{miss}^\top \\
&= d_{w,o,m}^{t+1} \boldsymbol{\Sigma}_{o,m}^{miss} - d_{w,o,m}^{t+1} \boldsymbol{\Sigma}_{o,m}^{miss.obs} (\boldsymbol{\Sigma}_{o,m}^{obs})^{-1} (\boldsymbol{\Sigma}_{o,m}^{miss.obs})^\top \\
&\quad + d_{w,o,m}^{t+1} \boldsymbol{\mu}_{o,m}^{miss} \boldsymbol{\mu}_{o,m}^{miss}^\top \\
&\quad + 2d_{w,o,m}^{t+1} \boldsymbol{\mu}_{o,m}^{miss} (\boldsymbol{\Sigma}_{o,m}^{miss.obs} (\boldsymbol{\Sigma}_{o,m}^{obs})^{-1} (\mathbf{X}_w^{obs} - \boldsymbol{\mu}_{o,m}^{obs}))^\top \\
&\quad + d_{w,o,m}^{t+1} \boldsymbol{\Sigma}_{o,m}^{miss.obs} (\boldsymbol{\Sigma}_{o,m}^{obs})^{-1} (\mathbf{X}_w^{obs} - \boldsymbol{\mu}_{o,m}^{obs}) \\
&\quad (\mathbf{X}_w^{obs} - \boldsymbol{\mu}_{o,m}^{obs})^\top (\boldsymbol{\Sigma}_{o,m}^{obs})^{-1} \boldsymbol{\Sigma}_{o,m}^{miss.obs}^\top \\
&\quad + d_{w,o,m}^{t+1} \boldsymbol{\mu}_{o,m}^{miss} \boldsymbol{\mu}_{o,m}^{miss}^\top - 2d_{w,o,m}^{t+1} \boldsymbol{\mu}_{o,m}^{miss} \boldsymbol{\mu}_{o,m}^{miss}^\top \\
&\quad - 2d_{w,o,m}^{t+1} \boldsymbol{\Sigma}_{o,m}^{miss.obs} (\boldsymbol{\Sigma}_{o,m}^{obs})^{-1} (\mathbf{X}_w^{obs} - \boldsymbol{\mu}_{o,m}^{obs}) \boldsymbol{\mu}_{o,m}^{miss}^\top \\
&= d_{w,o,m}^{t+1} \boldsymbol{\Sigma}_{o,m}^{miss} - d_{w,o,m}^{t+1} \boldsymbol{\Sigma}_{o,m}^{miss.obs} (\boldsymbol{\Sigma}_{o,m}^{obs})^{-1} (\boldsymbol{\Sigma}_{o,m}^{miss.obs})^\top \\
&\quad + d_{w,o,m}^{t+1} \boldsymbol{\Sigma}_{o,m}^{miss.obs} (\boldsymbol{\Sigma}_{o,m}^{obs})^{-1} \left(\boldsymbol{\Sigma}_w^{obs} + (\boldsymbol{\mu}_w^{obs} - \boldsymbol{\mu}_{o,m}^{obs}) \right. \\
&\quad \left. (\boldsymbol{\mu}_w^{obs} - \boldsymbol{\mu}_{o,m}^{obs})^\top \right) (\boldsymbol{\Sigma}_{o,m}^{obs})^{-1} \boldsymbol{\Sigma}_{o,m}^{miss.obs}^\top
\end{aligned}$$

References

1. http://wiki.ros.org/ar_track_alvar
2. Alizadeh, T.: Statistical learning of task modulated human movements through demonstration. Ph.D. thesis, Istituto Italiano di Tecnologia (2014)
3. Bishop, C.M., Lasserre, J., et al.: Generative or discriminative? getting the best of both worlds. *Bayesian statistics* **8**, 3–24 (2007)
4. Calinon, S.: Robot programming by demonstration. EPFL Press (2009)
5. Calinon, S.: A tutorial on task-parameterized movement learning and retrieval. *Intelligent Service Robotics* **9**(1), 1–29 (2016)
6. Calinon, S., Alizadeh, T., Caldwell, D.G.: On improving the extrapolation capability of task-parameterized movement models. In: *Intelligent Robots and Systems (IROS)*, 2013 IEEE/RSJ International Conference on, pp. 610–616. IEEE (2013)
7. Calinon, S., Li, Z., Alizadeh, T., Tsagarakis, N.G., Caldwell, D.G.: Statistical dynamical systems for skills acquisition in humanoids. In: *Humanoid Robots (Humanoids)*, 2012 12th IEEE-RAS International Conference on, pp. 323–329
8. Da Silva, B.C., Baldassarre, G., Konidaris, G., Barto, A.: Learning parameterized motor skills on a humanoid robot. In: *Robotics and Automation (ICRA)*, 2014 IEEE International Conference on, pp. 5239–5244. IEEE (2014)
9. Dempster, A.P., Laird, N.M., Rubin, D.B.: Maximum likelihood from incomplete data via the em algorithm. *Journal of the royal statistical society. Series B (methodological)* pp. 1–38 (1977)
10. Forte, D., Gams, A., Morimoto, J., Ude, A.: On-line motion synthesis and adaptation using a trajectory database. *Robotics and Autonomous Systems* **60**(10), 1327–1339 (2012)
11. Gams, A., Ijspeert, A.J., Schaal, S., Lenarčič, J.: On-line learning and modulation of periodic movements with nonlinear dynamical systems. *Autonomous robots* **27**(1), 3–23 (2009)
12. Ghahramani, Z., Jordan, M.I.: Supervised learning from incomplete data via an em approach. In: *Advances in neural information processing systems* 6. Citeseer (1994)
13. Herzog, D., Krüger, V., Grest, D.: Parametric hidden markov models for recognition and synthesis of movements. In: *BMVC*, vol. 8, pp. 163–172. Citeseer (2008)
14. Jordan, A.: On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. *Advances in neural information processing systems* **14**, 841 (2002)
15. Kober, J., Mülling, K., Krömer, O., Lampert, C.H., Schölkopf, B., Peters, J.: Movement templates for learning of hitting and batting. In: *Robotics and Automation (ICRA)*, 2010 IEEE International Conference on, pp. 853–858. IEEE (2010)
16. Lee, D., Ott, C.: Incremental kinesthetic teaching of motion primitives using the motion refinement tube. *Autonomous Robots* **31**(2-3), 115–131 (2011)
17. Lee, D., Ott, C., Nakamura, Y.: Mimetic communication model with compliant physical contact in human-humanoid interaction. *The International Journal of Robotics Research* **29**(13), 1684–1704 (2010)
18. Matsubara, T., Hyon, S.H., Morimoto, J.: Learning parametric dynamic movement primitives from multiple demonstrations. *Neural Networks* **24**(5), 493–500 (2011)
19. Pastor, P., Hoffmann, H., Asfour, T., Schaal, S.: Learning and generalization of motor skills by learning from demonstration. In: *Robotics and Automation, ICRA. IEEE International Conference on*, pp. 763–768 (2009)
20. Pastor, P., Kalakrishnan, M., Righetti, L., Schaal, S.: Towards associative skill memories. In: *Humanoid Robots (Humanoids)*, 2012 12th IEEE-RAS International Conference on, pp. 309–315. IEEE (2012)
21. Pastor, P., Righetti, L., Kalakrishnan, M., Schaal, S.: On-line movement adaptation based on previous sensor experiences. In: *Intelligent Robots and Systems (IROS)*, 2011 IEEE/RSJ International Conference on, pp. 365–371. IEEE (2011)

22. Pervez, A., Lee, D.: A componentwise simulated annealing em algorithm for mixtures. In: Joint German/Austrian Conference on Artificial Intelligence (Künstliche Intelligenz), pp. 287–294. Springer (2015)
23. Petersen, K.B., et al.: The matrix cookbook
24. Rasmussen, C.E., Nickisch, H.: Gaussian processes for machine learning (gpml) toolbox. *The Journal of Machine Learning Research* **11**, 3011–3015 (2010)
25. Rueckert, E., Mundo, J., Paraschos, A., Peters, J., Neumann, G.: Extracting low-dimensional control variables for movement primitives. In: Robotics and Automation (ICRA), 2015 IEEE International Conference on, pp. 1511–1518. IEEE (2015)
26. Schaal, S.: Dynamic movement primitives-a framework for motor control in humans and humanoid robotics. In: Adaptive Motion of Animals and Machines, pp. 261–280. Springer (2006)
27. Stulp, F., Raiola, G., Hoarau, A., Ivaldi, S., Sigaud, O.: Learning compact parameterized skills with a single regression. In: 13th IEEE-RAS International Conference on Humanoid Robots (Humanoids), pp. 417–422 (2013)
28. Ude, A., Gams, A., Asfour, T., Morimoto, J.: Task-specific generalization of discrete and periodic dynamic movement primitives. *Robotics, IEEE Transactions on* **26**(5), 800–815 (2010)
29. Ueda, N., Nakano, R.: Deterministic annealing em algorithm. *Neural Networks* **11**(2), 271–282 (1998)
30. Vijayakumar, S., Schaal, S.: Locally weighted projection regression: An $O(n)$ algorithm for incremental real time learning in high dimensional space. In: International conference on machine learning, proceedings of the sixteenth conference (2000)