Robotica: (2020) volume 38, pp. 457–474. © Cambridge University Press 2019
doi:10.1017/S026357471900078X

Compliant Parametric Dynamic Movement Primitives

Emre Ugur* and Hakan Girgin

Bogazici University, Computer Engineering, Istanbul, Turkey. E-mail: hakangirgin21@gmail.com

(Accepted April 25 2019. First published online: May 31, 2019)

SUMMARY
In this paper, we propose and implement an advanced manipulation framework that enables parametric learning of complex action trajectories along with their haptic feedback profiles. Our framework extends Dynamic Movement Primitives (DMPs) method with a new parametric nonlinear shaping function and a novel force-feedback coupling term. The nonlinear trajectories of the action control variables and the haptic feedback trajectories measured during execution are encoded with parametric temporal probabilistic models, namely parametric hidden Markov models (PHMMs). PHMMs enable autonomous segmentation of a taught skill based on the statistical information extracted from multiple demonstrations, and learning the relations between the model parameters and the properties extracted from the environment. Hidden states with high-variances in observation probabilities are interpreted as parts of the skill that could not be reliably learned and autonomously executed due to possibly uncertain or missing information about the environment. In those parts, our proposed force-feedback coupling term, which computes the deviation of the actual force feedback from the one predicted by the force-feedback PHMM, acts as a compliance term, enabling a human to scaffold the ongoing movement trajectory to accomplish the task. Our method is verified in a number of tasks including a real pick and place task that involves obstacles of different heights. Our robot, Baxter, successfully learned to generate the trajectory taking into the heights of the obstacles, move its end effector stiffly (and accurately) along the generated trajectory while passing through apertures, and allow human–robot collaboration in the autonomously detected segments of the motion, for example, when the gripper picks up the object whose position is not provided to the robot.

KEYWORDS: Learning from demonstration; Dynamic movement primitives; Hidden Markov models; Parametric actions; Haptic feedback models.

1. Introduction

While robots have been very effectively used for complex tasks in structured environments for decades (e.g., industrial robots), only recently they have started appearing in our daily life (e.g., mobile vacuum cleaners and autonomous cars). Recent advances in materials science, computer vision, machine learning and control with increasing computing power pave the way to the robots which can exhibit impressive skills in uncertain and dynamic environments. The era of "Cambrian Explosion for robotics"[1] is probably approaching due to the developments in these fronts. One avenue toward progress lies in semi-structured domains that limit the scope of the robot's operation to a predictable subset, and where useful robot functionality can be achieved by limited levels of autonomy. Supermarkets, warehouses, and to some extent office and hospital environments provide such domains where the physical layout and structures are well-defined; the set of objects and their common locations are mostly fixed; and the required tasks such as pick and place are typical. Endowing robots with limited but effective set of skills that exploit multi-modal sensory feedback to the maximum extent can overcome a major obstacle to deployment of real manipulation systems in

* Corresponding author. E-mails: emre.ugur@boun.edu.tr

supermarkets and similar, semi-structured domains. Exploitation of the sensory feedback can be used in perceiving the action possibilities provided to the robot,[2] in generating plans based on detected actions and their predicted effects,[3] in monitoring action execution, and in intelligent and robust response to unexpected external perturbations.[4]

In the semi-structured environments mentioned about, the set of required skills for well-defined tasks can be effectively transferred to the robots using learning from demonstration (LfD) framework.[5] From possibly several demonstrations of a particular skill in different configurations, the robot needs to capture the important characteristics from its observations, and be able to reproduce the learned skill in new configurations, under perturbation, noise or uncertainty. Action executions that achieve a particular task in such predictable environments would have typical characteristics: they follow common movement trajectories and they generate regular perceptual changes in the environment. While the overall shape is common in general, the details of the movement trajectories might differ due to constraints imposed by the environment; that is, different motions might be required to achieve the same task in different settings. If the robot has the ability to perceive the properties of the environment that affect the execution of an action, it can learn the relations between these properties and the action control variables. In such situations, the robot should be able to parameterize the movement trajectory based on these properties, predict the perceptual changes, and successfully execute the actions even in novel environment configurations. However if the robot cannot find a correlation between any perceived/given property of the environment and the movement trajectory, it would have no means to explain the variance observed in successful action executions. Such a variance might be due to unimportance of following an exact trajectory during the corresponding part of the execution or due to robot's inability to extract (or learn from) the related properties of the environment. In the former case, the robot can safely apply any trajectory from the observed range, whereas in the latter case we expect that the robot decides that the corresponding part of the action cannot be reliably learned and executed autonomously. In other words, external help would be required during robot execution. Therefore, it is plausible to design the system such that in both cases, the robot automatically switches to a mode that follows an average trajectory, complying with external forces during execution of the corresponding part of the skill.

In this paper, we propose a manipulation framework, Compliant Parametric Dynamic Movement Primitives (CPDMP), that can encode and generate complex movement trajectories, parameterize these trajectories based on specific properties of the environment, learn sensory feedback models observed during action executions, and automatically switch to a mode that enables external help during observed high-variance segments of the demonstrated skill. The compliance is realized through exploiting the difference between the actual and expected sensory feedback, which is computed from learned sensory feedback models.

Dynamic Movement Primitives (DMPs)[6] are used as the base system and are extended to encode and reproduce the required actions. DMPs encode the demonstrated trajectory as a set of differential equations, and offers advantages such as one-shot learning of nonlinear movements, real-time stability and robustness under perturbations with guarantees in reaching the goal state, generalization of the movement for different goals, and linear combination of parameters.[7] As noted by Colome and Torras,[8] in order to learn the relations between trajectories and parameter space, a large number of parameters are required to be explored with the original DMP formulation. In this paper, we used Parametric Hidden Markov Models (PHMM) to encode the nonlinear part of the trajectory. PHMM allows us to learn from multiple demonstrations, automatically segment the observed skill based on the statistical information, learn the relations between these segments and the environment properties if possible, and provide variance information that can be used to deduce the certainty of the learned model for the corresponding segment. We further encoded the observed force-feedback trajectories using PHMMs, built force-feedback models for each learned skill and exploited these models, adding a new feedback term to DMP to achieve compliance when needed. Our parametric model was first verified in a simulated obstacle avoidance task and then in a real cabinet opening task. Finally, it was used in a pick and place task with a real robot that involves obstacles of different heights. The robot successfully learned how to generate the trajectory based on the heights of the obstacles, in which parts of the execution it requires to accurately follow the generated trajectory, and finally to automatically allow human–robot collaboration exploiting the learned haptic feedback models. After providing a brief overview of the related work in the next section, we will present the details of our proposed model and the experiments performed in Sections 3 and 4.

2. Related Work

LfD[5] has been suggested as an efficient and intuitive way to teach new skills to the robots, where the robot observes, learns and imitates the actions demonstrated by the human tutors. LfD has been applied to various robotic learning problems including object grasping and manipulation.[4,9–12] Among others, learning methods that are based on dynamic systems[6] and statistical modeling (with GMMs[9] and HMMs[13]) have been popular in the recent years.

DMPs[6] encode the demonstrated trajectory as a set of differential equations, and offer advantages such as one-shot learning of nonlinear movements, real-time stability and robustness under perturbations with guarantees in reaching the goal state, generalization of the movement for different goals, and linear combination of parameters. The parameters of the system can be learned with different advanced algorithms such as Locally Weighted Regression[14] and Locally Weighted Projection Regression.[15] Statistical modeling techniques generally use probabilistic models such as Gaussian Mixture Models[9,16] or temporal probabilistic models such as Hidden Markov Models[13] to encode the motion from multiple demonstrations. They learn the important features of the demonstrated motions and model the statistical regularities within the spatial data. Encoding the statistical properties, for example, variation and correlation information in the movements, has been proved to be useful, for example, in modulating the system based on various constraints such as kinematic and task based ones.[16] Despite such advantages, statistical methods do not directly provide robustness, generalization and guaranteed convergence to goal position as in DMPs.

In DMP models, the nonlinear function encodes the shape of the trajectory whereas the other terms makes the trajectory attain a goal position from an initial position within a desired duration. Therefore, DMP can respond robustly to the static or dynamic changes of the movements' initial or goal positions while preserving its specific shape in simple tasks reaching to an object while avoiding an obstacle. However, in more complex tasks such as opening the door of a cabinet from different handles or pouring from a container with amounts of water, only preserving the shape of the demonstrated trajectory might not generate a trajectory capable of accomplishing the task. In such cases, we need to create trajectory models that depend on some parameters extracted from the environment such as the position or weight of the objects.

Zhou et al.[17] used a simple 2D obstacle avoidance task to show that standard DMP approach is not designed to learn from multiple trajectories and therefore cannot encode the important parts of multiple demonstrations. In this task, a motion that brings the end effector from one side of an circular obstacle to the other side is taught to the robot. The shape of the trajectories with a number of different goal positions is shown and a DMP model for these trajectories is extracted. When the goal position of the DMP model is changed to a new goal position that is roughly symmetrically on the other side of the initial position, the created trajectory fails to avoid the obstacle. To deal with this problem, a parametric representation that relates the obstacle size to the action should be learned. Zhou et al. developed a method that performs LWR by defining a new objective function dependent on the environment restrictions and parameters, and that uses a model-switching algorithm to be able to train over all the parametric space. While this is an effective algorithm for the corresponding task, we would like to combine the advantages of DMP-based approaches with the information provided by statistical modelling techniques such as the variance in the demonstrations. Ude et al.[18] used LWR and GPR methods to compute new DMP-based trajectories given parameters that describe the characteristics of the action, such as the goal point. Their method could be generalized with complex trajectories for both periodic and discrete movements. We would like to learn the relation between the trajectories and any related aspect of the environment. Pervez et al.[19] learned a separate GMM for each demonstration and mixed the separately learned GMMs to generate trajectories for new task parameter values. Instead, we learn one base HMM and a weighted additive term that changes the mean of the HMM Gaussians given the task parameters. In addition, we exploited the variance information to let the system to enable human–robot collaboration in this paper.

In order to enable physical collaboration, the robot was required to learn haptic feedback models that encode the default predicted force feedback measured during execution of the corresponding skill. Memorized force and tactile profiles have already been successfully utilized in modulating learned DMPs in difficult manipulation tasks that contain high degrees of noise in perception such as grasping and in-hand manipulation of objects from incorrect positions or flipping boxes using chopsticks.[20,21] However, we believe that rather than memorizing one single haptic profile for a skill, learning general multi-model sensory models might provide us with more generalizable and robust manipulation skills. In this paper, we modeled the haptic feedback trajectories with PHMMs and used those models in an additional coupling term similar to ref. [20]. Chu et al. also learned such multi-modal models based on Hidden Markov Models from temperature, pressure and fingertip information for exploratory object classification tasks;[22] however, the learned models were not used to adapt any further action execution. Latent Dirichlet Allocation[23] and recently deep networks[24] were used to learn multi-modal models from different sensory information such as temperature, pressure, fingertip, contacts, proprioception and speech; however, these models were used only to categorize the sensory data without any effect on action execution. More recently, Kramberger et al. investigated the same problem of generalization of force/torque profiles for contact tasks.[25] In their work, these profiles are modeled by Locally Weighted Regression (LWR) which has local generalization capabilities, hence successful at intermediate query points.

CPDMP can learn the nonlinear shaping terms dependent on a specific parameter of the environment and execute the predicted trajectory of a new environmental configuration by taking into account the variances, switching back and forth its compliance mode. In CPDMP, the nonlinear terms are learned by PHMM.[26] To the best of our knowledge, PHMM was used once in robotics applications for a liquid pouring task to create a model that links the joint space and sensory feedback information of the robot to the amount of the liquid.[27] However, their work consisted of learning the trajectory itself by PHMM and did not address robust execution to dynamical changes. We additionally exploited the covariance matrices of PHMM in order to capture the uncertainty in the corresponding part of the learned skill and in order to allow human–robot interaction.

3. Methods

In this section, we will first describe Dynamical Movement Primitives (DMP) method, its default formulation as a set of differential equations, how complex movement trajectories are encoded with a nonlinear shaping function, and how this function might be learned from demonstration. Next, we will provide the details of our proposed method, CPDMP, where we propose a parametric version of the shaping function that also encodes the statistical information in the movement trajectory of multiple demonstrations of the learned action. Additionally, we add a force-feedback coupling term that enables the robot to learn the force/torque trajectory expected to be measured during execution of the movement trajectory, and to exploit it for reacting toward unexpected perturbations.

3.1. Dynamic movement primitives

DMPs can be interpreted as linear spring systems perturbed by external forcing terms. For one degree of freedom, DMP is composed of the following set of differential equations:

$$\tau\dot{v} = K(g-x) - Dv - K(g-x_0)s + Kf(s) \tag{1}$$

$$\tau\dot{x} = v \tag{2}$$

where the movement starts from x_0 position and ends at goal position g. x, $\dot{x}$, and $\dot{v}$ correspond to the position, velocity and acceleration of the system. The changes in velocity and acceleration are scaled with a constant τ. K and D are the proportionality and damping constants, respectively. D is set to $2\sqrt{K}$ to make the system critically damped. $f(s)$ is a nonlinear function that encodes the shape of the trajectory. Finally, s is a phase variable that monotonically decreases and converges at value zero:

$$\tau\dot{s} = -\alpha s \tag{3}$$

where α is a constant representing the convergence rate of the phase variable from 1 to 0.

The first three terms in Eq. (1) allows the system to act as a spring damper system, guaranteeing to reach to the goal position and/or tracking a moving goal. The influence of $f(s)$ function depends on the value of s and its influence saturates as s approaches to zero. Therefore, the system is guaranteed to reach to the goal position independent of how $f(s)$ encodes the nonlinear trajectory or even if the goal position changes in the beginning or during action execution. The system also preserves its shape when the initial and final positions change or the system is perturbed during execution. Finally, one can make sure simultaneous evaluation of multiple degrees of freedoms or multiple systems by integrating the same phase variable in the corresponding DMPs in parallel.

Nonlinear Shaping Function: $f(s)$ shaping function is a nonlinear function that encodes an arbitrarily complex trajectory of a demonstrated movement. To learn the movement, $x(t)$ is recorded. Next, $v(t)$ and $\dot{v}(t)$ are calculated for each time step t. $s(t)$ is calculated using an appropriately set τ value. Finally, Eq. (1) is re-arranged so that the value of $f(s)$ is calculated as "target $f(s)$" using $s(t)$, $x(t)$, $v(t)$, and $\dot{v}(t)$:

$$f_{\text{target}}(s) = -(g - x) + (g - x_0)s + \frac{\tau\dot{v} + Dv}{K} \tag{4}$$

The problem of learning of the movement trajectory is now transformed to learning of the function $f(s)$ using the sample points calculated above.

Locally Weighted Regression: The $f(s)$ function can be encoded as the weighted sum of m radial basis functions:

$$f(s) = \frac{\sum_i^m w_i \psi_i(s) s}{\sum_i \psi_i(s)} \tag{5}$$

where $\sum_i \psi_i(s)$ is a normalizing term. ψ_i corresponds to the ith radial basis function in the following form:

$$\psi_i = exp(-h_i(s - c_i)^2) \tag{6}$$

where c_i and h_i are the center and the bandwidth of the corresponding basis function, respectively. The multiplicative term, the phase variable s, ensures that the influence of the nonlinear shaping function vanishes toward the end of the movement, ensuring reaching to the target position. The learning problem can be simplified by setting centers and bandwidths manually. In this case, the centers are logarithmically distributed between 0 and 1, and bandwidths are set to values inversely proportionally to the centers as follows: $h_i = m/(c_i^2)$. The weights $\{w_1, w_2, \ldots w_m\}$ that encode the shaping function of each particular demonstrated movement, on the other hand, can be learned by linear regression. The matrix form of these weights and the nonlinear function for one demonstration are as follows:

$$\mathbf{f} = \begin{bmatrix} f_{\text{target}}(s_1) \\ \cdots \\ f_{\text{target}}(s_T) \end{bmatrix}, \mathbf{w} = \begin{bmatrix} w_1 \\ \cdots \\ w_N \end{bmatrix} \tag{7}$$

$$\mathbf{X} = \begin{bmatrix} \frac{\psi_1(s_1)}{\sum_i^N \psi_i(s_1)} s_1 & \cdots & \frac{\psi_N(s_1)}{\sum_i^N \psi_i(s_1)} s_1 \\ \cdots & \cdots & \cdots \\ \frac{\psi_1(s_T)}{\sum_i^N \psi_i(s_T)} s_T & \cdots & \frac{\psi_N(s_T)}{\sum_i^N \psi_i(s_T)} s_T \end{bmatrix} \tag{8}$$

Based on the linear relationship between these weights, pseudo-inverse of matrix $\mathbf{X}$ is taken and multiplied with $\mathbf{f}$ to find the weights:

$$\mathbf{X}\mathbf{w} = \mathbf{f} \tag{9}$$

$$\mathbf{w} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{f} \tag{10}$$

3.2. Our proposal: Compliant Parametric Dynamic Movement Primitives (CPDMP)

In this paper, we extend the original DMP as follows:

- We model the nonlinear shaping function with a parametric temporal probabilistic method that can encode the trajectory of the same movement primitive executed several times. For this we use PHMM (Section 3.2.1).
- We integrate a coupling term that enables the system to react to unexpected perturbations that can be measured by force/torque sensors (Section 3.2.2).

3.2.1. Nonlinear shaping function with PHMM. We extend the nonlinear shaping function to a parametric form using a PHMM-based parametric model instead of weighted sum of radial basis functions as follows:

$$\tau \dot{v} = K(g - x) - Dv - K(g - x_0)s + Kf(\theta_m, s) \tag{11}$$

where

- θ_m corresponds the set of parameters that affect the movement trajectory, and
- $f(\theta_m, s)$ corresponds to the new nonlinear shaping function that takes θ_m as input parameter along with the phase variable s.

A standard HMM λ of N hidden states is composed of the prior distributions π_i, the transition probabilities a_{ij} and the observation probability distributions b_i :

$$\lambda = \{\pi_i, a_{ij}, b_i\}_{i,j=1}^{N}$$

For continuous data, observation distribution b_i is a single multi-variate Gaussian distribution, $b_i(\boldsymbol{x}) = \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)$, for each hidden state i. In this model of a single trajectory, some part of the trajectory is covered by each Gaussian, and state index i is incremented along with the evolving trajectory. When multiple trajectories are encoded, each Gaussian encodes the variance of the corresponding part of the training data.

PHMM extends HMM by including a global parametric variation in the observation probabilities of the HMM states. The mean of the Gaussian distribution of each hidden state is calculated taking into account the given parameter. In a linear model, the mean of the Gaussian is calculated by adding the weighted linear sum of the parameters $\boldsymbol{\theta}$ to a baseline mean:

$$\boldsymbol{\mu}_i(\boldsymbol{\theta}) = \boldsymbol{W}_i\boldsymbol{\theta} + \bar{\boldsymbol{\mu}}_i \tag{12}$$

where $\boldsymbol{\theta}$ is the parameter vector that might include one or more parameters, $\bar{\boldsymbol{\mu}}_i$ is the baseline mean of the Gaussian for the observation probability distribution of the hidden state i, and $\boldsymbol{W}_i$ is the weight matrix. As an example, given a number of obstacle avoidance demonstrations, if the observation is 3D position trajectory of the end effector and the parameters are the height and width of an obstacle, $\boldsymbol{\theta}$ vector would be 2; $\bar{\boldsymbol{\mu}}_i$ and $\boldsymbol{\mu}_i(\boldsymbol{\theta})$ vectors would be 3 and $\boldsymbol{W}_i$ matrix would be 3×2 sized. Learning a PHMM model corresponds to learning the weights ($\boldsymbol{W}_i$) and the baseline means ($\bar{\boldsymbol{\mu}}_i$) of each hidden states in addition to standard HMM parameters such as the prior probabilities of the hidden states (π_i) and the transition probabilities between states (a_{ij}).

Expectation Maximization algorithm described in ref. [28] is used for training. In the expectation step, forward–backward algorithm[29] is used to compute the probability $\boldsymbol{\gamma}_{t,i} = P(q_t = i|\boldsymbol{x}, \lambda)$, being in the hidden state i at time t given the entire sequence of $\boldsymbol{x}$ and the current parameters (λ) of the PHMM. In the maximization step, the PHMM parameters, that is π_i, a_{ij}, $\boldsymbol{W}_i$ and $\bar{\boldsymbol{\mu}}_i$, are updated. While the first two parameters are updated following the standard maximization step of the EM algorithm (see ref. [28] for details), the weight matrix and base mean vectors are updated in a group as in ref. [26]. For this, two new variables are defined for each hidden state i:

$$\boldsymbol{Z_i} = [\boldsymbol{W_i} \quad \bar{\boldsymbol{\mu}}_i] \qquad\qquad \boldsymbol{\Omega_k} = \begin{bmatrix} \boldsymbol{\theta}_k \\ \mathbf{1} \end{bmatrix}$$

such that $\boldsymbol{\mu}_i(\boldsymbol{\theta}) = \boldsymbol{Z_i}\boldsymbol{\Omega_k}$. Only $\boldsymbol{Z_i}$ is updated using each observation sequence $\boldsymbol{x_k}$ and the corresponding $\boldsymbol{\Omega_k}$ where k stands for the index of the observation:

$$\boldsymbol{Z_i} = \left[\sum_{k,t} \boldsymbol{\gamma}_{k,t,i}\boldsymbol{x}_{k,t}\boldsymbol{\Omega}_k^T\right]\left[\sum_{k,t} \boldsymbol{\gamma}_{k,t,i}\boldsymbol{\Omega}_k\boldsymbol{\Omega}_k^T\right]^{-1}$$

Once the means are updated, the covariance matrices of the Gaussians for the hidden states are updated in the standard way:

$$\Sigma_{\mathbf{i}} = \sum_{\mathbf{k,t}} \frac{\boldsymbol{\gamma}_{\mathbf{k,t,i}}}{\sum_{\mathbf{t'}} \boldsymbol{\gamma}_{\mathbf{k,t',i}}}(\boldsymbol{x}_{\mathbf{k,t}} - \boldsymbol{\mu}_{\mathbf{i}}(\boldsymbol{\theta}_{\mathbf{k}})(\boldsymbol{x}_{\mathbf{k,t}} - \boldsymbol{\mu}_{\mathbf{i}}(\boldsymbol{\theta}_{\mathbf{k}}))^{\mathbf{T}}$$

In this EM iterative method, the parameters of the PHMM are initialized by first training a global non-parametric HMM using the same EM method. The only difference in the global case $\boldsymbol{\theta}_k$ is set to $\mathbf{1}$. To initialize the means of the global HMM, we used k-means algorithm.

Note that in this formulation, the parameter $\boldsymbol{\theta}$ only affects the mean, and it is assumed that the variance of the observation probability distribution does not depend on this parameter.

When the nonlinear functions of DMPs are learned by PHMM through a dataset that is composed of demonstrations, each having at least one parameter, we obtain a parametric model of the nonlinear function. Given a novel configuration of the environment, the parameter from this situation is extracted and is given to the PHMM model to compute the new means of the Gaussians of the hidden states. As a result, parametric DMPs adapting to the changing environments are created.

Movement Generation. Gaussian Mixture Regression (GMR) method[30] is used to compute the movement trajectory. When a new parameter is given to the model retrieved by PHMM, each hidden state produces new multivariate Gaussian distributions. The mean vector of these distributions $\boldsymbol{\mu}_i$ and the covariance matrix $\boldsymbol{\Sigma}_i$ can be expressed as partitioned matrices by splitting the input $\boldsymbol{x}$ and output $\boldsymbol{y}$ as

$$\boldsymbol{\mu}_i = \begin{bmatrix} \boldsymbol{\mu}_i^x \\ \boldsymbol{\mu}_i^y \end{bmatrix} \qquad \boldsymbol{\Sigma}_i = \begin{bmatrix} \boldsymbol{\Sigma}_i^{xx} & \boldsymbol{\Sigma}_i^{xy} \\ \boldsymbol{\Sigma}_i^{yx} & \boldsymbol{\Sigma}_i^{yy} \end{bmatrix} \tag{13}$$

According to the GMR,[30] the output vector can be found by inserting the input vector and the Gaussian distribution acquired from the PHMM as follows:

$$\boldsymbol{y} = \sum_{i=1}^{N} h_i[\boldsymbol{\mu}_i^y + \boldsymbol{\Sigma}_i^{yx}(\boldsymbol{\Sigma}_i^{xx})^{-1}(\boldsymbol{x} - \boldsymbol{\mu}_i^x)] \tag{14}$$

Here, h_i are the weights of the marginal distribution of the input and are calculated as follows:

$$h_i = \frac{\mathcal{N}(\boldsymbol{x};\, \boldsymbol{\mu}_i^x,\, \boldsymbol{\Sigma}_i^{xx})}{\sum_{i=1}^{N} \mathcal{N}(\boldsymbol{x};\, \boldsymbol{\mu}_i^x,\, \boldsymbol{\Sigma}_i^{xx})} \tag{15}$$

where $\mathcal{N}(\boldsymbol{x};\, \boldsymbol{\mu}_i^x,\, \boldsymbol{\Sigma}_i^{xx})$ is the multivariate Gaussian density function of the input.

With these values, Eqs. (1) and (2) are used to calculate the change in position and the velocity and these changes are reflected to generate the corresponding trajectory.

3.2.2. Compliance in parametric DMPs. In order to react to external forces, we further extended the PDMP by integrating a force-feedback coupling term. PHMM-based shaping function and force-feedback coupling term are incorporated in the DMP formulation as follows:

$$\tau \dot{v} = K(g - x) - Dv - K(g - x_0)s + Kf(\theta_m, s) + \zeta(\theta_f, s) \tag{16}$$

where

- θ_f corresponds to the set of parameters that affect the measured force/torque feedback during execution of the movement, and
- $\zeta(\theta_f, s)$ corresponds to the force-feedback coupling term that enables the system to react to unexpected external forces based on inputs θ_f and s.

In order to react to unexpected external forces, the system requires to learn the forces that are expected to be observed during execution of the corresponding primitive, and encode these observed forces as desired forces ($F_{\text{des}}(\theta_f, s)$). During execution, the system computes the difference between actual and desired forces, generates an error based on this difference, and exploits this error to react to the perturbation in different ways. The system might try to minimize the error or allow the external forces to affect the movement depending on the task. PHMMs are used to encode desired forces that are denoted by $F_{\text{des}}(\theta_f, s)$. A general form of the proposed coupling term, which uses a parametric function of θ_f as opposed to the non-parametric averaging method proposed by Pastor et al.[20], is given as follows:

$$\zeta(\theta_\mathbf{f}, \mathbf{s}) = \beta\ \mathbf{K_1 J_{sensor}^T K_2 (F - F_{des}}(\theta_\mathbf{f}, \mathbf{s})) \tag{17}$$

With this formulation, an error is generated by calculating the difference between the current force $\mathbf{F}$ and the desired force $\mathbf{F_{des}}$ in task-space, that is, the 6D wrench space of the end-effector. This error is reflected to the movement using $\mathbf{K_1}$ and $\mathbf{K_2}$, which are positive definite gain matrices in task space and joint space, respectively; and $\mathbf{J^T_{sensor}}$, which is the transpose of the Jacobian with respect to sensors by which the forces are measured. β parameter is used to change the way the system reacts to the external forces. Note that both $f(\theta_m, s)$ and $F_{des}(\theta_f, s)$ are encoded as parametric temporal probabilistic models with PHMMs that are detailed in Section 3.2.1.

Depending on the task, the system can be configured to react to the external forces in different ways. β parameter and K_1 and K_2 coefficients should be adjusted for this. In this section, we will describe two different ways of reacting to external forces:

Minimize Force-Feedback Error to Correct Mistakes in The Movement Trajectory. The robot, while executing its action, might make small mistakes in generating its movement trajectory. For example, in an action which involves reaching in order to grasp an object, the target position might be set incorrectly due to noise in perception. Then, the force feedback measured while touching and grasping the object would be different from the expected one. In such cases, the learned force-feedback trajectory models can compute how much the movement deviated from the desired one, and how to correct the movement based on this deviation. For this, the force-feedback coupling term ζ can be set as follows:

$$\zeta = 1\ \mathbf{K_1 J^T_{sensor} K_2 (F - F_{des})} \tag{18}$$

where β was set to 1, and $\mathbf{K_1}$ and $\mathbf{K_2}$ are set manually depending on the task. We already studied such correcting actions in ref. [31] where our system exploited those models to correct the perturbed movements during executions with the aim of generalizing to novel configurations.

Exploit Force-Feedback Error to Comply with The External Forces. In some tasks, the interference of an external agent, that is a human or another robot, might be required for successfully achieving the goal. In such cases, rather than minimizing the force-feedback error as above, the system can exploit the error to compute the signals provided from outside, and change the ongoing movement trajectory complying with these signals. In such a scenario, the force-feedback coupling term ζ should be set as follows:

$$\zeta = -\mathbf{K_1 J^T_{sensor} K_2 (F - F_{des})} \tag{19}$$

where β was set to -1, and $\mathbf{K_1}$ and $\mathbf{K_2}$ are set to identity matrices.[1]

Such a setting practically brings the robot to compliant mode. In our study, we would like the system to learn which parts of the movement require compliance and which parts require stiff movement autonomously. This information is designed to be captured directly from demonstration. If part of the demonstrated movement always follows the same path, in other words the movement has low variance, the robot can generate the trajectory and follow it accurately. On the other hand, if one part of the demonstrated movement follows different paths, that is the corresponding segment of the movement has high variance, the robot might conclude that there is high uncertainty in the corresponding part and triggers compliance mode automatically to allow external help.

Therefore, we propose to use the variance information provided in some parts of the demonstrations as a measure of compliance for the corresponding part. When the variances are high, we expect the robot to behave more compliant, so that the human can intervene and help the robot to complete its task. As mentioned before, PHMMs can automatically segment trajectories taking into account the local variance information, and can provide the required variance information free of charge. Therefore, each hidden state i in the nonlinear shaping function $f(\theta_m, s)$ that generates movement trajectory is associated with a discrete compliance value, and the compliance coefficient β is be set accordingly:

$$\beta_i = \begin{cases} -1 \text{ if } |\Sigma_i^{jk}| > \sigma, \exists\, j, k \in \{x, y, z\} \\ 0 \text{ o.w.} \end{cases} \tag{20}$$

[1]While the influence of both forcing function and force-feedback term will asymptotically vanish toward the end of the movement, our framework does not provide any stability guarantees due to integration of Eq. (19). In the experiments, we observe that the generated movement were stable around the region of the demonstration instances.

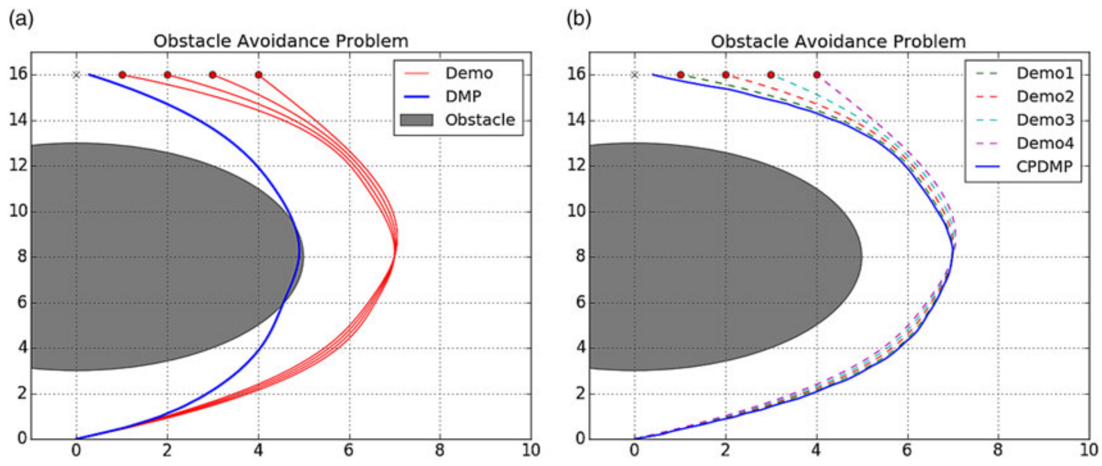

Fig. 1. (a) Trajectories that are produced for a single end point close the one shown with a cross, with a DMP model learned from the trajectories going to four different end positions. Even though all the demonstrations avoid the obstacle, newly produced trajectory cannot avoid the obstacle. (b) The PDMP model that is learned with the end positions of the demonstrations succeeded to avoid the obstacles, no matter how close they are to the single point shown with the cross.

where β in Eq. (17) is set to β_i during generation of the ith segment and σ corresponds to a variance threshold. In other words, in case any component of covariance matrix Σ_i exceeds the threshold σ_t, the system decides to run in compliant mode in ith hidden state, exploiting the error computed from force-feedback model.

4. Experiments

In this section, after showing a proof-of-concept verification of our method in a 2D simulated obstacle avoidance task (Section 4.1), we will provide the generalization capability of our method with a real robot, first in response to change in target position (Section 4.3), and then in response to other changes in the environment (Section 4.4).

4.1. Target position change in simulated obstacle avoidance task

In this section, we show the advantage of our method (CPDMP) over standard DMP in its capability of preserving the shape of the trajectory while generalizing the trajectory to different target positions. A simulated 2D obstacle avoidance task[17] is used for this purpose. The aim in this task is to learn avoiding from an obstacle while moving from an initial position to an arbitrary target position. In Fig. 1, the gray ellipse shows the obstacle, (0,0) position is the initial position of the movement, and the red round points on the top correspond to the targets of the movement. The system is provided one demonstration trajectory (red line) for each red target point. The x-coordinate of the end position is used as the θ_m parameter and a PHMM model is trained to learn $f(\theta_m, s)$ in the CPDMP formulation, Eq. 11. The force-feedback coupling term is not considered in this task. After learning the movement from these multiple trajectories, the system is requested to generate a new trajectory for a target point that is placed to the left side of the observed target points. In order to compare CPDMP with the goal-point generalization capability of DMPs, a DMP model (Eq. 1) was also trained from these demonstrations. The weights of the radial basis functions (Eq. 5) were learned using LWR taking into account the forcing terms of all four trajectories. As shown in Fig. 1(a), the trajectory learned by DMP fails to encode the important part of the demonstrations. The non-parametric DMPs are not originally designed to learn from multiple trajectories with different goal points; therefore, LWR probably learned an average representation from the trajectories of the shaping functions. Therefore, while roughly preserving the shape of the movement, it failed to avoid the obstacle. On the other hand, CPDMP, when parameterized with the position of the target point, can successfully avoid the obstacle. The blue line in Fig. 1(b) shows that CPDMP avoids the target successfully, keeping sufficient distance from the target.

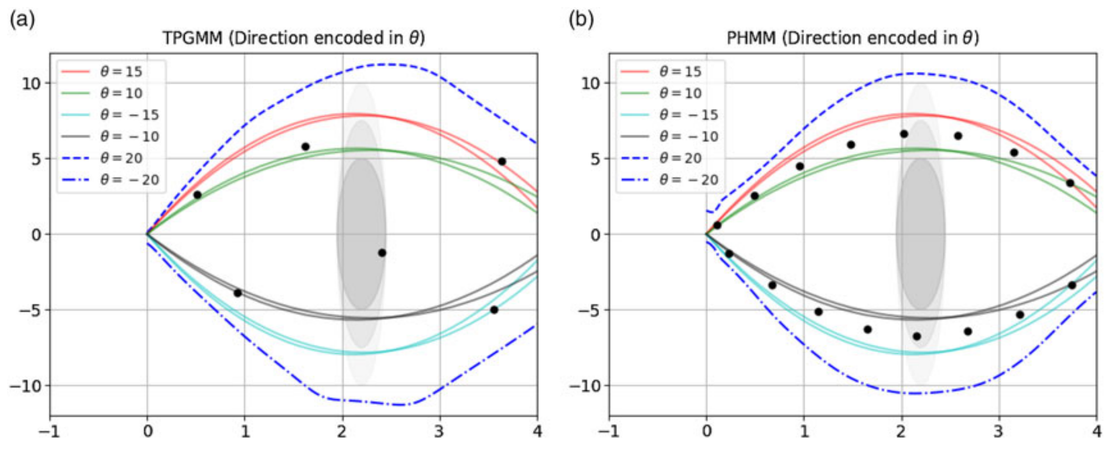

Fig. 2. PHMM and TPGMM models were trained with the solid lines and the corresponding parameters provided in the legend. The magnitude and the sign of the parameter (θ) encodes the width of the obstacle and the direction of the demonstration. The Gaussian means are shown with the black dots. As shown, both models could produce trajectories that can pass from both sides of a new obstacle of width of 20 units.

4.2. Comparison with TPGMM

In the previous subsection, we showed that CPDMP can deal better with demonstrations with different goal positions compared to the non-parametric DMPs. In this subsection, we compare the capabilities of the underlying PHMM representation of CPDMP with a parametric representation that also encodes the statistical information of multiple trajectories, namely Task Parameterized Gaussian Markov Mixteure Models (TPGMM).[32] Assume that two sets of trajectories that pass around obstacles of two different sizes are provided to the system in a 2D setting. Half of each set of the trajectories traverses from one side of the obstacle and the other half passes around the other side.

In the first experiment, both the width of the object and from which side the traversal occurs were provided as the parameter along with each trajectory. Note that we focus only the trajectory encoding and generation with PHMM and TPGMM; therefore, we will refer to the parameter with variable θ. As shown in Fig. 2, both PHMM and TPGMM produced trajectories that can pass from both sides of a new obstacle of larger width.[2]

In the second experiment, direction of the demonstration was not provided to the system and therefore only the width of the obstacle was encoded in the parameter of the corresponding demonstration. TPGMM in this case failed to generate trajectories for the new parameters as all the Gaussians were used to reproduce the trajectories with GMR (Fig. 3a). In the PHMM reproduction, in order to benefit from the state transition probabilities, we implemented a simple heuristic that extracts the disjoint chains of states starting from the start state, and provided the Gaussians of the chains separately in GMR trajectory reproduction phase. As shown in Fig. 3(b), this approach allowed PHMM to produce trajectories that passed from both sides of the new obstacle[4].

[2]The (0,0) and (2,θ) positions are set as the first and the second candidate frame of references in training TPGMM. Note that six Gaussians were used in TPGMM training as the TPGMM EM algorithm implemented in the Programming by Demonstration Library[3] could not encode the trajectories with higher number of Gaussians. While more smooth trajectories could have been generated by TPGMM with higher number of Gaussians, our focus in this analysis is not related to smoothness but it is rather about the capability of generating trajectories based on given demonstrations and parameters.

[3]http://www.idiap.ch/software/pbdlib/

[4]Note that one can attempt to extract the state chains from the learned TPGMM with additional steps. For this, clustering that takes into account means and variances of the Gaussians can be applied to find set of Gaussians that are close to each other. However the result would highly depend on the clustering algorithm, where another pass of hyper-parameter tuning (the number of clusters) model tuning (the type of the clustering method) should be applied. Another way to extract the chains from the learned TPGMM is to estimate the pairwise transition frequencies by counting the number of transitions between the corresponding Gaussians; and use these estimates to form chains. While such an approach that first finds the Gaussians and then estimates the corresponding transition probabilities can provide similar results in this particular example, TPHMM is a method that particularly serves this purpose in a principled way: through co-joint optimization of the transition probabilities and the Gaussian means and variances.

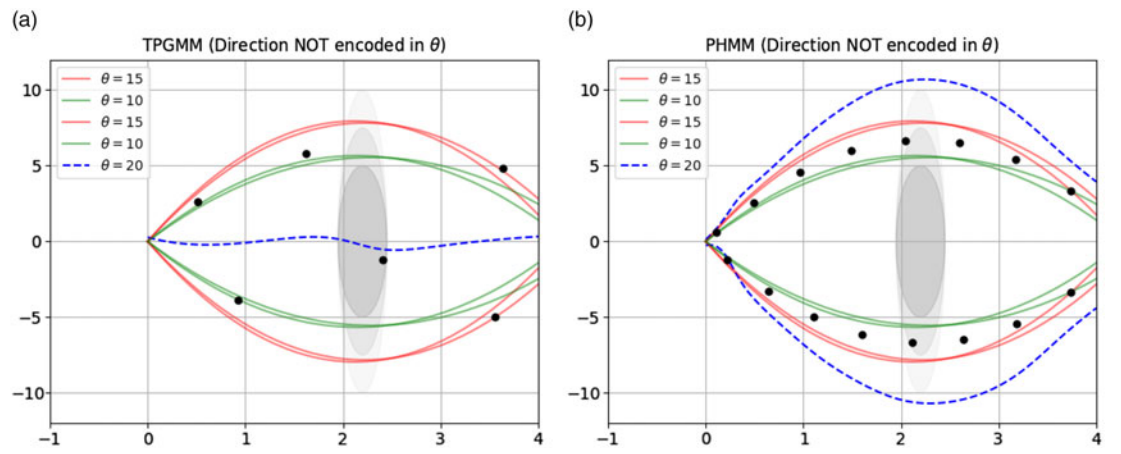

Fig. 3. PHMM and TPGMM models were trained with the solid lines and the corresponding parameters provided in the legend. The Gaussian means are shown with the black dots. The parameter (θ) encoded only the width of the obstacle. As shown, only PHMM could produce trajectories that can pass from both sides of a new obstacle of width of 20 units.

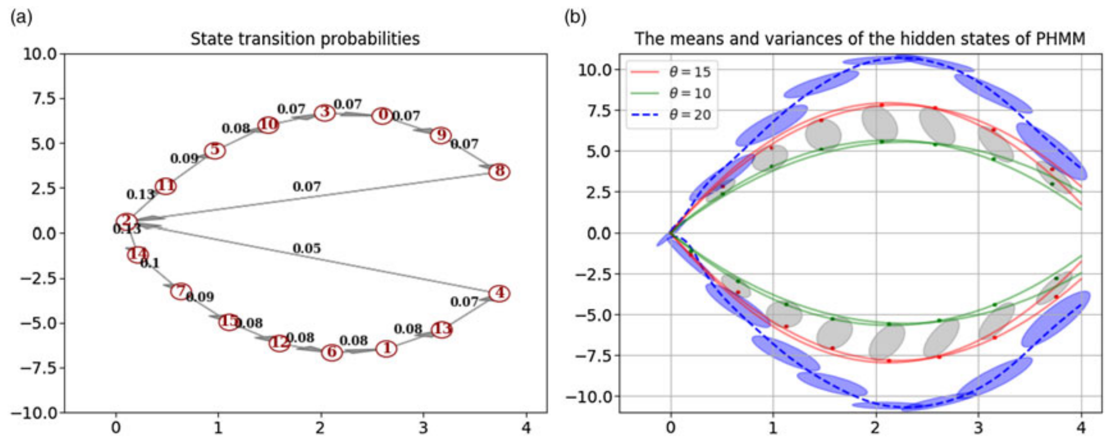

Fig. 4. Analysis of PHMM training, encoding and reproduction. (a) The self-transition probabilities are not displayed for clarity. Transition probabilities from 4th and 8th states to the initial state (2nd state) are non-zero as the eight trajectories were concatenated back-to-back for PHMM learning. (b) Solid and dashed lines correspond to the demonstrations and reproductions, respectively. Gray ellipses, red/blue dots and blue ellipses correspond to initial HMM variances, final parameterised means and final PHMM variances, respectively.

The state transition probabilities encoded in the PHMM are provided in Fig. 4(a). The self-transition probabilities were not displayed in the figure for clarity. As shown, starting from the hidden state 2, two different chains were generated, and the Gaussians of the corresponding hidden states were provided separately in GMR trajectory reproduction. Figure 4(b) shows the variances of the initial non-parametric HMM by the gray ellipses, the means of the Gaussians for different parameters with green and red dots, and the variances of the Gaussians of the final PHMM with blue ellipses.

4.3. *Handle change in cabinet opening task with Baxter robot*

Our experimental setup consists of Baxter robot which has two 7 degrees of freedom and a cabinet. The door of the cabinet is attached to the cabinet with a hinge joint, and five different handles are placed in a row on the door with 5 cm intervals. As shown in Fig. 5, a 3D printed hook is mounted on the wrist of the left arm of the robot and clamped to one of the handles in the beginning of the movement. The cabinet opening skill was demonstrated to the robot by kinesthetic teaching, that is, by physically moving the end-effector of the robot. The position of the handle relative to the robot base in the lateral axis is used as the θ_m parameter and a PHMM model is trained to learn $f(\theta_m, s)$ in the CPDMP formulation, Eq. 11. The force-feedback coupling term is not considered in

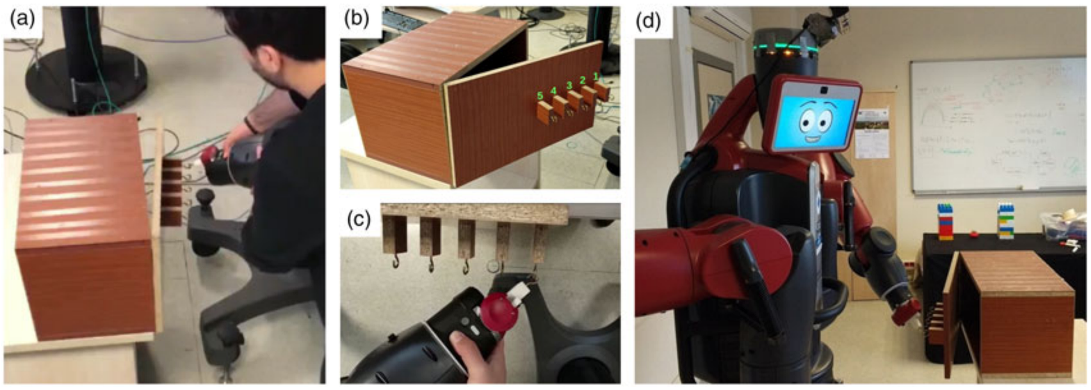

Fig. 5. The cabinet opening task (a) provides a snapshot from kinesthetic teaching, (b) shows the handles attached to the door of the cabinet, (c) provides a top–down snapshot of the demonstration of the cabinet opening movement from the first handle, and (d) gives a snapshot from Baxter's autonomous execution.

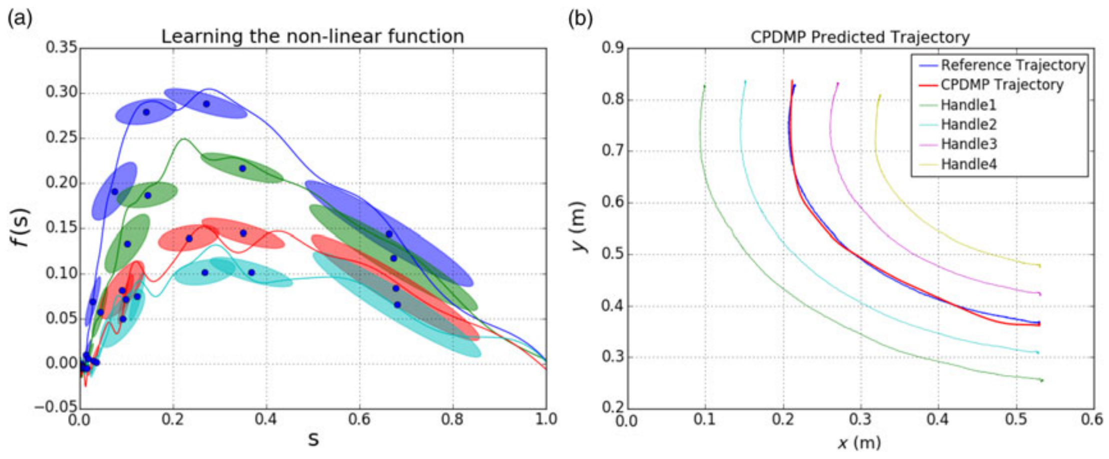

Fig. 6. (a) The shaping function of the trajectories taken from all handles and learned PHMM model. The radius and center of the ellipses are, respectively, expressing the center and variance of the Gaussian density functions produced by the hidden factors. (b) The graph of the trajectories in $x - y$ plane to open the cabinet door with not demonstrated third handle produced with the CPDMP. The trajectory produced with CPDMP follows a similar path to the reference trajectory.

this task. Demonstrations with all handles were used for learning except with the one in the middle. In each demonstration, 3D position trajectory of the end effector in the task space was stored along with the corresponding time information. Seven different hidden states were used, and K is set to an empirically found value, 5000.

Figure 6(a) shows the centers of the density functions of each hidden state produced by CPDMP. As shown the centers are related to the shaping function of the demonstrated trajectories based on the given parameter, the position of the handle. As mentioned before, in this paper, the covariance matrices of the Gaussian functions are found by applying non-parametric HMM method to the complete data, and are set to the same values for each Gaussian function of the corresponding hidden state. With this learned model, the novel position of the third handle, which was not included in the training set, is given as the parameter to the robot. New centers of the hidden states are calculated for this parameter. As shown in Fig. 6(b), CPDMP could generate a trajectory that matches well with the reference movement. When the generated trajectory is executed by the Baxter robot, the cabinet was successfully opened from the third handle. Figure 7 shows a number of snapshots from Baxter's performance and provides the link to the corresponding video.

4.4. Pick and place an object avoiding obstacles, allowing human–robot interaction

Our final experiment is designed to evaluate the full range of the capabilities of our proposed method. With this experiment, we aim to verify the capabilities of our system in parameterizing the movement

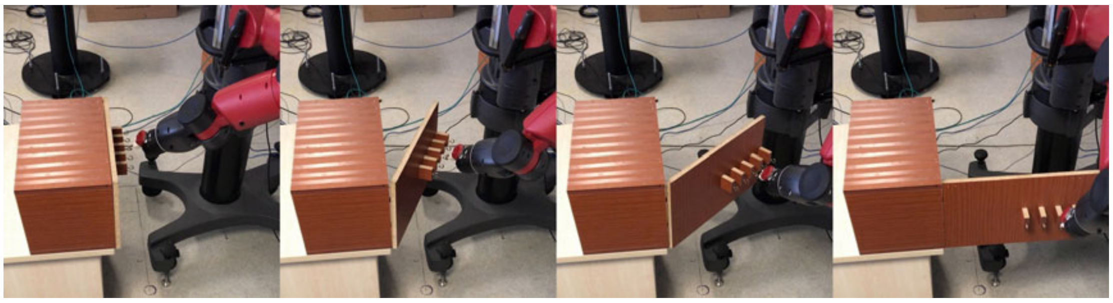

Fig. 7. The application of the trajectory produced by the learned PDMP model that is given initial position of the 3rd handle as start pose as parameter. See the video link : https://youtu.be/GD1WFIIBCsA

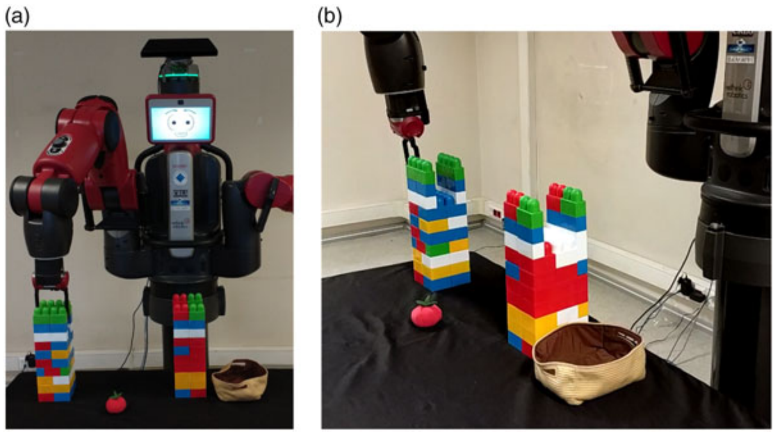

Fig. 8. Baxter and the pick and place task. The gripper passes through the first narrow aperture, picks up the red object, passes through the second wider aperture and places the object on the table.

trajectory based on environment properties and exploiting the encoded variances observed in different segments of the movement trajectory for compliant control.

Setup: Experimental setup consists of two U-shaped toy blocks with varying heights, a soft round toy, a basket and the Baxter robot as shown in Fig 8. The wrench measurements at the end-effector of the Baxter are provided by the built-in system of the robot through integrating individual joint efforts. The pick and place task requires that the robot's gripper starts moving from one side of the table (from the right side from the perspective of the robot), passes through the narrow space of the first block taking into account the height of the block, picks up the round toy, and passes through a wider space of the second block taking into account its height as well, finally placing the toy into the basket. While the initial and final position of the robot gripper are always same and the gripper should go through the apertures without collision, the object to be picked up can be placed in various positions between the two blocks. This task simulates situations where the object position cannot be reliably obtained and cannot be explicitly communicated to the robot, for example, when the object is inside an opaque bag. In such situations, a human intervention is necessary to locate the object and help the robot to pick up the object by physically bringing the gripper over the object. We would like the robot to autonomously detect this from the unexplainable high-variance observed in the corresponding segment of the motion. Moreover, the height of the blocks might be different and this information should also be taken into account during learning and executing the skills.

The pick and place skill is taught to the robot in three different configurations, that is, with three different block heights. In each configuration, the two blocks have the same heights, and the round toy is placed to three different positions to simulate the uncertainty. Figure 9 shows these configurations. The elevation of the gripper (position along z-axis) for these trajectories are shown in Fig. 10 with dashed lines. As shown, each trajectory starts from the same initial position and ends at the same final position. The gripper is elevated to certain points depending on the height of the blocks, as visible

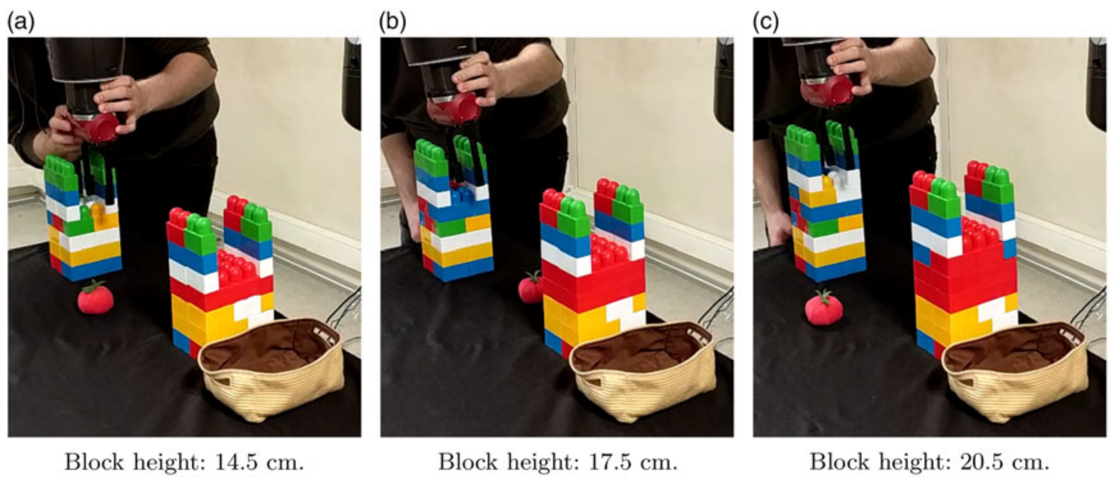

Fig. 9. Setups for demonstration of the pick and place task.

from their values during [5–10] and [20–25] s. The gripper is brought close to the table at around second 15 to pick-up the toy. Figure 11 shows the lateral movement (along the x-axis) of the gripper during demonstrations with dashed lines. As shown, the trajectories of the nine demonstrations along the x-axis are very close to each other except when the object was picked up from a different position each time.

Our system is required to learn the force-feedback model of the corresponding pick and place skill. Therefore, the trajectories that were obtained during kinesthetic teaching were replicated by the robot without human intervention. The force-feedback trajectories that were measured during these replications were used to train the PHMM force-feedback models. θ_f parameter was set to a fixed value in the training as the differences in the noisy wrench readings of the Baxter robot obtained from different obstacle heights or object weights were not expected to have a significant effect on this task.

The height of the block was used as the θ_m parameter of the CPDMP that learns a general movement model from the provided nine demonstrations. The position of the object is not provided to the system. The compliance threshold σ is empirically set to 1.0 in Eq. (20). Our CPDMP model is required to both encode the trajectories based on the height of the blocks and learn the movement segments that exhibit high variance during object pick up.

Results: The CPDMP model, after learning from the demonstrated nine trajectories, is requested to produce a trajectory in a new configuration where the height of the blocks was set to a new height: 23.5 cm. The z-component of the generated trajectory is shown with the solid line in Fig. 10. The results show that the system can generate a trajectory that moves the end effector through apertures avoiding collision with the blocks. Furthermore, the variance in the movement of the predicted trajectory is high at the pick-up location, and low while passing through the apertures of the blocks.

We further examined the details of the generated trajectory. The solid line in Fig. 11 shows the predicted trajectory, and the ellipses show the predicted means and variances along with their corresponding eigenvalues in red boxes. Recall that the gaussian ellipses correspond to the observation probability distributions of the nonlinear term. Their means are shifted up/down in the figure, keeping the time coordinate constant, in order to be visualized clearly and to be compared with the trajectory itself. The variances of the Gaussians in the x-direction, hence the eigenvalues in that direction, are higher in the regions where the demonstration variance was high; that is, objects were picked up from different locations, resulting in autonomously switching to compliant mode.

We executed the learned pick and place behavior with the novel block height where the object is placed to an arbitrary position. The robot was able to complete the task successfully: the movement trajectory adapted to the height of the block; the gripper automatically remained stiff and followed the learned trajectory accurately while passing through the apertures, and switched to the compliant mode during object pick-up. Remaining stiff while passing through the narrow gap was important and required as the noisy force/torque sensors of the Baxter might read erroneous external forces that

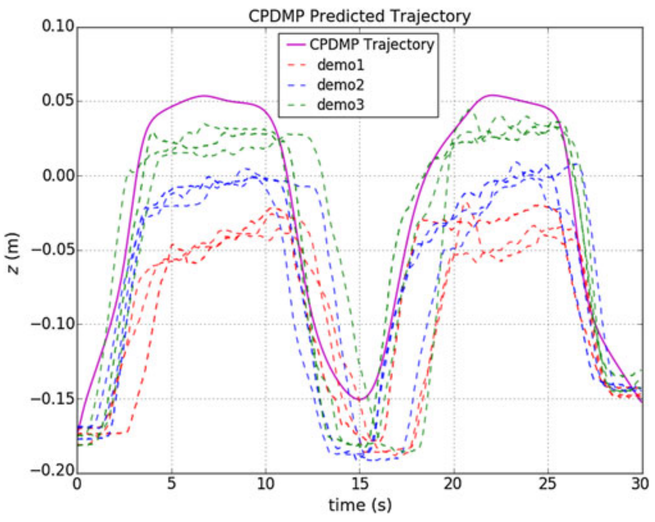

Fig. 10. The elevation change in the gripper during kinesthetic teaching and autonomous execution. The dashed lines correspond to the position of the gripper along the *z*-axis obtained during pick and place demonstrations. Three different demonstrations were provided for each of the training environment. The solid line corresponds to the movement trajectory produced by the CPDMP method in the environment with blocks of novel height.

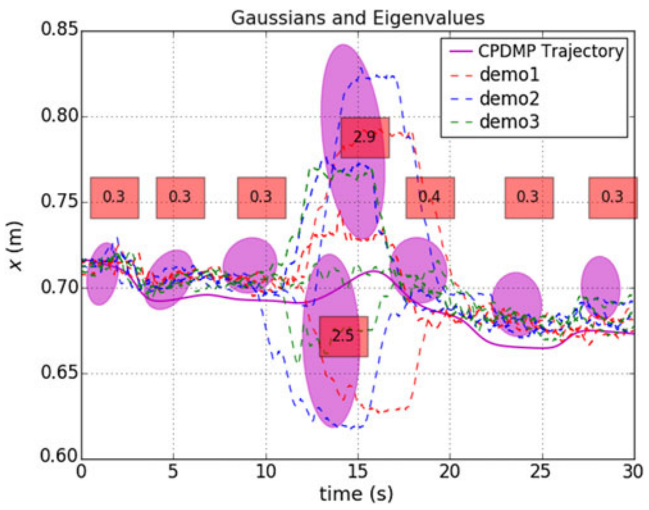

Fig. 11. The lateral movement of the gripper during kinesthetic teaching and autonomous execution. The dashed lines correspond to the position of the gripper along the *x*-axis obtained during pick and place demonstrations. Three different demonstrations were provided for each of the training environment. The solid line corresponds to the movement trajectory produced by the CPDMP method in the environment with blocks of novel height. The ellipses correspond to the observation probability distributions of the 8 hidden factors. As shown, in the initial phase of the movement (when the gripper passes through the first aperture) the model start with small variances, during object pick-up that corresponds to the fourth and fifth hidden factors the variances become larger, and in the rest of the movement the variances become smaller again.

would result in inaccurate movement due to the force-feedback coupling term, and causing collision with the blocks. On the other hand, when the robot switched to the compliant mode between 12 and 17th s due to large variance, the force-feedback coupling term that generates an error from the difference between predicted and measured forces changed the path of the movement toward the force exerted by the human. This allowed the human to control the robot gripper (only) during that period and scaffold action of the robot: The human placed the gripper over the red object and the object is successfully picked up by the gripper. After 17th s, as the variance of the corresponding segment was low, the robot automatically switched back to stiff mode, which allowed the gripper to pass through the second aperture, placing the object to the target position (see the execution snapshots and the link to the corresponding video in Fig. 12).

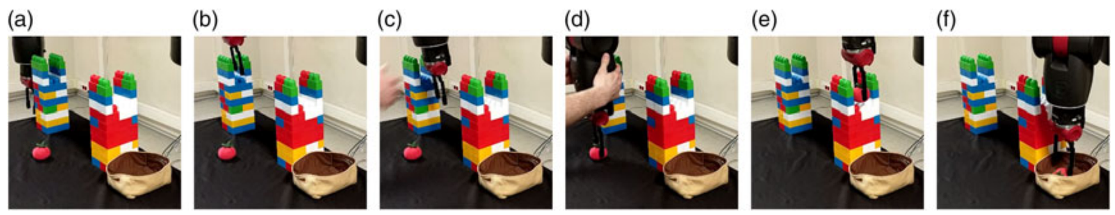

Fig. 12. Execution of the learned pick-and-place skill with CPDMP. The object is placed to an arbitrary position. (a) Initial gripper position. (b) The gripper follows the learned trajectory accurately in the stiff mode due to the low variance of the hidden states encoded in HMM (see the first three hidden states in Fig. 11) also adapting to the novel height of the block. (c) The robot switches to compliant mode, enabling the DMP force-feedback coupling term that computes the error between predicted and actual forces and moves the end-effector in the direction of the external forces. (d) A human physically now can scaffold the movement of the end effector while it is in the compliant mode, and positions it so that the red object can be picked up. (e) The robot switches back to its stiff and accurate movement mode. (f) The object is placed to the target position, inside the basket. See the video link : https://youtu.be/VACw5_DNQeU

5. Conclusion

In this paper, we proposed an advanced manipulation framework that learns complex action trajectories along with their haptic feedback profiles. Our framework extends the DMP method with a new parametric nonlinear shaping function and a novel force-feedback coupling term. The trajectories of the movement control variables and the force/torque measurements are encoded with parametric temporal probabilistic models, namely PHMMs. PHMMs enable autonomous segmentation of a taught skill based on the statistical information extracted from multiple demonstrations, and learning the relations between the model parameters and the properties extracted from the environment. Hidden states with high variances in observation probabilities are interpreted as parts of the skill that could not be reliably learned and autonomously executed due to possibly uncertain or missing information about the environment. In those parts, our proposed force-feedback coupling term, which exploits the deviation of the actual force feedback from the one predicted by the force-feedback PHMM, acts as a compliance term, enabling a human to scaffold the ongoing movement trajectory to accomplish the task. Our method is verified in a number of tasks including a real pick and place task that involves obstacles with different heights. Our robot, Baxter, successfully learned to generate the trajectory taking into account the heights of the obstacles, move its end effector stiffly (and accurately) along the generated trajectory while it passes through apertures, and allow human–robot collaboration in the autonomously detected segments of the motion, for example, when the gripper picks up the object whose position is not provided.

The compliance in the pick and place task is not related and does not benefit from any parametric relation in the environment, and therefore the force-feedback model was not trained with any particular parameter. As an example, if the end-effector had carried a heavy object that significantly changed the measured forces when the robot was required to be compliant, the weight of the objects should have been used as the parameter to the model. In a previous study,[31] we showed the advantage of using the weight of the object as the parameter to the force-feedback model in an object pushing task. While we had used PHMMs to encode the force-feedback trajectory, the robot was provided only one trajectory to learn: the movement of the end-effector along a straight line while pushing an object. For this, we had encoded the movement with a non-parametric model, namely normalized sum of radial basis functions as in standard DMPs, whereas in the proposed framework, we used a parametric formulation for encoding the movement, and exploited the variance in the hidden states to trigger compliance.

Our PHMM-based trajectory encoding and GMR-based trajectory reproduction approach was shown to be effective to extrapolate outside the demonstrated range of a single parameter. On the other hand, PHMMs in general allow trajectory encoding based on a combination of linear relations of parameter and hidden states for multiple parameters. In our pick-and-place experiment, one can imagine including different heights for the pairs of obstacles, and the location of the object to the parameter set. We plan to investigate learning more complex relations that allow more complex tasks in the future. Additionally, we plan to investigate learning of nonlinear relations between environment properties and the centers of PHMMs through exploiting various kernels, and study methods that also combine the covariance matrices.[33] While learning the environment–action relations, the properties of the environment, such as the position of the handle or the height of the block, are directly provided to our model. While perceiving these properties is not in the scope of this paper and is mostly straightforward using computer vision techniques, selection of the relevant set of features from possibly infinite number of features is a challenging and open problem. Physical simulators can be used to systematically change different parts of the environment and verify if the features extracted from the corresponding part have any relation with the action. This in turn requires the capability to fully model the environment, which is not very difficult with the current technology just from the perception of the environment. The aim of our H2020 project IMAGINE (https://imagine-h2020.eu) is to enable robots to understand the structure of their environment and how it is affected by its actions through integrating generative models and physics simulators.

Acknowledgements
This research has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement no. 731761, IMAGINE; and was partially supported by TUBITAK BIDEB under agreement No 117C016 and by Bogazici Resarch Fund (BAP) Startup project no 12022.

References

1. G. A. Pratt, "Is a cambrian explosion coming for robotics?" *J. Econ. Persp.* **29**(3), 51–60 (2015).
2. J. J. Gibson, *The Ecological Approach to Visual Perception* (Lawrence Erlbaum Associates, Hillsdale, New Jersey, 1986).
3. E. Ugur, E. Oztop and E. Sahin, "Goal emulation and planning in perceptual space using learned affordances," *Rob. Auto. Syst.* **59**(7–8), 580–595 (2011).
4. P. Pastor, L. Righetti, M. Kalakrishnan and S. Schaal, "Online Movement Adaptation Based on Previous Sensor Experiences," *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, San Francisco, CA (IEEE, 2011) pp. 365–371.
5. B. D. Argall, S. Chernova, M. Veloso and B. Browning, "A survey of robot learning from demonstration," *Rob. Auto. Syst.* **57**(5), 469–483 (2009).
6. S. Schaal, "Dynamic Movement Primitives-a Framework for Motor Control in Humans and Humanoid Robotics," **In**: *Adaptive Motion of Animals and Machines* (H. Kimura, K. Tsuchiya, A. Ishiguro, H. Witte, eds) (Springer, Tokyo, 2006) pp. 261–280.
7. P. Pastor, M. Kalakrishnan, S. Chitta, E. Theodorou and S. Schaal, "Skill Learning and Task Outcome Prediction for Manipulation," *2011 IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai (IEEE, 2011) pp. 3828–3834.
8. A. Colomé and C. Torras, "Dimensionality Reduction and Motion Coordination in Learning Trajectories with Dynamic Movement Primitives," *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Chicago (IEEE, 2014) pp. 1414–1420.
9. S. Calinon, P. Evrard, E. Gribovskaya, A. Billard and A. Kheddar, "Learning Collaborative Manipulation Tasks by Demonstration Using a Haptic Interface," *2009 International Conference on Advanced Robotics (ICAR)*, Munich (IEEE, 2009) pp. 1–6.
10. T. Asfour, P. Azad, F. Gyarfas and R. Dillmann, "Imitation learning of dual-arm manipulation tasks in humanoid robots," *Int. J. Hum. Rob.* **5**(02), 183–202 (2008).
11. H. B. Amor, O. Kroemer, U. Hillenbrand, G. Neumann and J. Peters, "Generalization of Human Grasping for Multi-fingered Robot Hands," *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Algarve, Portugal (IEEE, 2012) pp. 2043–2050.
12. M. Mühlig, M. Gienger and J. J. Steil, "Interactive imitation learning of object movement skills," *Auto. Rob.* **32**(2), 97–114 (2012).
13. D. Lee and C. Ott, "Incremental kinesthetic teaching of motion primitives using the motion refinement tube," *Auto. Rob.* **31**(2–3), 115–131 (2011).
14. C. G. Atkeson, A. W. Moore and S. Schaal, "Locally Weighted Learning for Control," **In**: *Lazy Learning* (David W. Aha ed.) (Springer, Dordrecht, 1997) pp. 75–113.
15. S. Vijayakumar and S. Schaal, "Locally Weighted Projection Regression: Incremental Real Time Learning in High Dimensional Space," *Proceedings of the Seventeenth International Conference on Machine Learning* (Morgan Kaufmann Publishers Inc., Massachusetts, United States, 2000) pp. 1079–1086.
16. S. Calinon, F. Guenter and A. Billard, "On learning, representing, and generalizing a task in a humanoid robot," *Part B Cyber. IEEE Trans. Syst. Man Cyber.* **37**(2), 286–298 (2007).
17. Y. Zhou and T. Asfour, "Task-Oriented Generalization of Dynamic Movement Primitive," *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Vancouver (2017) pp. 3202–3209.
18. A. Ude, A. Gams, T. Asfour and J. Morimoto, "Task-specific generalization of discrete and periodic dynamic movement primitives," *IEEE Trans. Rob.* **26**(5), 800–815 (2010).
19. A. Pervez and D. Lee, "Learning task parameterized dynamic movement primitives using mixture of gmms," *Int. Service Robot*. (2017). http://elib.dlr.de/113356/

20. P. Pastor, M. Kalakrishnan, L. Righetti and S. Schaal, “Towards Associative Skill Memories,” *2012 12th IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, Osaka, Japan (IEEE, 2012) pp. 309–315.
21. P. Pastor, H. Hoffmann, T. Asfour and S. Schaal, “Learning and Generalization of Motor Skills by Learning from Demonstration,” *IEEE International Conference on Robotics and Automation, 2009 (ICRA’09)*, Kobe, Japan (IEEE, 2009) pp. 763–768.
22. V. Chu, I. McMahon, L. Riano, C. G. McDonald, Q. He, J. Martinez Perez-Tejada, M. Arrigo, N. Fitter, J. C. Nappo, T. Darrell and K. J. Kuchenbecker, “Using Robotic Exploratory Procedures to Learn the Meaning of Haptic Adjectives,” *2013 IEEE International Conference on Robotics and Automation (ICRA)*, Kobe, Japan (IEEE, 2013) pp. 3048–3055.
23. T. Araki, T. Nakamura, T. Nagai, K. Funakoshi, M. Nakano and N. Iwahashi, “Autonomous Acquisition of Multimodal Information for Online Object Concept Formation by Robots,” *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, San Francisco, CA (IEEE, 2011) pp. 1540–1547.
24. A. Droniou, S. Ivaldi and O. Sigaud, “Deep unsupervised network for multimodal perception, representation and classification,” *Rob. Auto. Syst.* **71**, 83–98 (2015).
25. A. Kramberger, A. Gams, B. Nemec, D. Chrysostomou, O. Madsen and A. Ude, “Generalization of orientation trajectories and force-torque profiles for robotic assembly,” *Robot. Auton. Syst.* **98**, 333–346 (2017). https://doi.org/10.1016/j.robot.2017.09.019
26. A. D. Wilson and A. F. Bobick, “Parametric hidden Markov models for gesture recognition,” *IEEE Trans. Pattern Anal. Mach. Intell.* **21**(9), 884–900 (1999).
27. L. Rozo, P. Jiménez and C. Torras, “Force-Based Robot Learning of Pouring Skills Using Parametric Hidden Markov Models,” *2013 9th Workshop on Robot Motion and Control (RoMoCo)*, Poland (IEEE, 2013) pp. 227–232.
28. L. R. Rabiner, “A tutorial on hidden markov models and selected applications in speech recognition,” *Proc. IEEE* **77**(2), 257–286 (1989).
29. S. J. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach* (Pearson Education Limited, Harlow, United Kingdom, 2016), chapter 15.
30. Z. Ghahramani and M. I. Jordan, “Supervised Learning from Incomplete Data via an em Approach,” **In**: *Advances in Neural Information Processing Systems 6* (J. D. Cowan, G. Tesauro, J. Alspector, eds) (Morgan Kaufmann, San Francisco, CA, 1994) pp. 120–127.
31. H. Girgin and E. Ugur, “Associative Skill Memory Models,” *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (2018) pp. 6043–6048.
32. S. Calinon, “A tutorial on task-parameterized movement learning and retrieval,” *Intell. Serv. Rob.* **9**(1), 1–29 (2016).
33. A. K. Tanwani and S. Calinon, “Learning robot manipulation tasks with task-parameterized semitied hidden semi-Markov model,” *IEEE Rob. Auto. Lett.* **1**(1), 235–242 (2016).