Variable Duration Movement Encoding with Minimal Intervention Control

Martijn J.A. Zeestraten[1], Sylvain Calinon[2,1], Darwin G. Caldwell[1]

***Abstract*— Programming by Demonstration (PbD) offers a user-friendly way to transfer skills from human to robot. Typically, demonstration data do not contain the control inputs required to reproduce the demonstrated skill. These can be obtained from a low-level controller that tracks the modeled movement. We present a PbD approach for minimal intervention control — a control strategy that only corrects perturbations that interfere with task performance. The novelty of our approach is the probabilistic encoding of the movement duration, providing a performance measure that enables minimal intervention control in a temporal sense. This is achieved by combining a probabilistic movement encoding based on Hidden Semi-Markov Model (HSMM) with Model Predictive Control (MPC). The probabilistic model is used to construct an objective function, hereby assuming that variance is a measure for task performance. The proposed method is demonstrated in a robot experiment and compared with our earlier work.**

I. INTRODUCTION

Fulfilling the prospect of robots, leaving factory floors to enter the human world and act among us, could lie decades away. However, the transition of robots from large scale factory plants, as can be found in the car manufacturing industry, towards smaller manufacturers might lie around the corner.

Programming by Demonstration (PbD) is a learning technique that could accelerate this transition [1]. It provides a user-friendly programming solution that allows users to program a task by showing a number of successful task executions. The technique is ideal for small manufactures, that are characterized by shorter production life cycles, since PbD allows fast (re)programming and does not require expert programmers.

One of the challenges in PbD is to find a suitable way to model and reproduce the demonstrated skill. There are two common ways to represent a demonstrated skill: autonomous and non-autonomous.

Autonomous representations model a movement as a dynamical system; encoding a time-independent relation between the dynamical features of the movement (e.g. position, velocity and acceleration). Such systems form an attractor landscape in which the goal state is a global minimum. Reproduction of the encoded skill is achieved by following the steepest descent of the attractor landscape [2], [3].

[1]Istituto Italiano di Tecnologia - Via Morego 30 - 16163 Genova, Italy martijn.zeestraten@iit.it

[2]Idiap Research Institute - Rue Marconi 19 - CH-1920 Martigny, Switzerland

The research leading to these results has received funding from the People Programme (Marie Curie Actions) of the European Unions Seventh Framework Programme FP7/2007-2013/ under REA grant agreement no 608022.

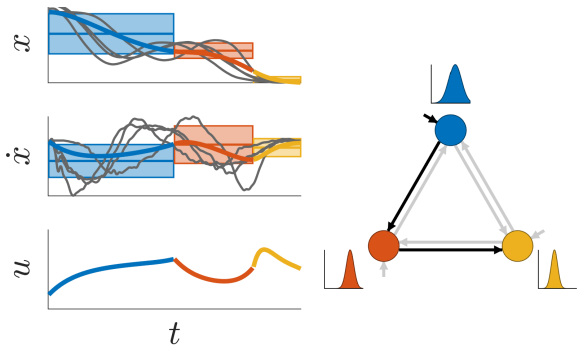


Fig. 1: 1D example of the proposed approach. Position x and velocity $\dot{x}$ (gray) — obtained by demonstration — are encoded in three Gaussian kernels (colored rectangles: The center line indicates mean, outer lines indicate one standard deviation). The colored lines visualize the position and velocity profiles of the reproduction. The bottom left graph shows the control signal u that is used to obtain the reproduction. It is obtained by solving an optimal control problem. The required cost function is based on a state sequence that is synthesized using the state transition and duration information of a HSMM (right). The colors of the control signal u, position x and velocity $\dot{x}$, correspond to the colors of the activated kernels.

Non-autonomous movement representations encode an explicit dependency between a temporal signal and the dynamical features of the movement. The retrieval of the movement from the model is driven by this temporal signal that can represent time directly, or indirectly using a decay term (see e.g. [4]–[6]). We call these systems non-autonomous since the system evolution depends on a variable that is not part of the system state.

Because autonomous systems are not driven by a temporal signal, they provide an intrinsic robustness to perturbations that distort the temporal evolution of the system state. When the robot encounters an obstacle, the interaction force will remain constant since the action of the robot only depends on the system state (which remains constant). Non-autonomous systems would generate undesirable interaction forces in such situations due to the evolution of the temporal signal. In contrast, temporal dependency can be beneficial when the temporal features are part of the task performance; e.g. tasks that require synchronization with automated systems.

Reproduction of the movement on a robot requires a low-level controller with an adaptive control strategy. We follow the *minimal intervention principle* for the controller design. This principle states that *"Deviations from the average trajectory are only corrected when they interfere with task performance"* [7]. This results in a strategy that trades off

control effort and task performance.

The combination of probabilistic movement models with optimal control provides a promising way to learn minimal intervention controllers from demonstration [8], [9]. These approaches assume that the required task performance can be related to the covariance encoded in the probabilistic model; large variance in the dynamical features indicates that accurate tracking is not required and vice versa. These methods, however, do not consider variability in the total movement duration. Hence, the control strategy does not follow a minimal intervention strategy in a temporal sense.

We propose a PbD approach for minimal intervention control that considers both spatial and temporal variability. Similarly to spatial task performance, we assume that accuracy required to perform a task can be extracted from variability apparent among demonstrations. Tasks with strict temporal requirements will show low temporal variability, whereas tasks without these requirements will show higher variability. We want to exploit this information to create a non-autonomous movement representation that allows modification of its temporal behavior depending on the task requirements.

Similarly to our previous work [8], the reproduction of the encoded task will be based on a minimal intervention controller that is learned from demonstration. The novelties of this work are three-fold: (i) instead of solving the optimal control problem for the complete duration of the task, we follow a Model Predictive Control (MPC) approach where the control commands are recomputed at each time step for a receding horizon; (ii) we show that a minimal intervention controller can be obtained from a Gaussian Mixture Model (GMM) by replacing the smooth reference, generated by time-based regression, with a piecewise reference only consisting of the Gaussian centers and covariances; (iii) we show that we can obtain a controller with variable temporal behavior by encoding the movement in a Hidden Semi-Markov Model (HSMM), and use this model to synthesize the control objective online.

This paper is structured as follows: In Section II, we describe the components of the proposed method. A comparison with a time driven method is given in Section III. Section IV shows an application of the Task-Parameterized HSMM (TP-HSMM) in a robotic experiment using a Barrett WAM. Discussion and future work is presented in Sections V and VI, respectively.

II. METHOD

We distinguish two phases in the approach: the *demonstration* phase and the *reproduction* phase. During the demonstration phase, the robot is provided with N demonstrations of the skill. Each demonstration consists of T_n datapoints $\boldsymbol{\xi}_t = [\boldsymbol{x}_t^\top, \dot{\boldsymbol{x}}_t^\top]^\top$ where $n \in \{1, \cdots, N\}$, and $\cdot^\top$ indicates the matrix transpose. Unlike other non-autonomous movement encoding, we do not consider an explicit temporal signal. We only assume that the data are sampled with a regular time interval Δt. After demonstration, the obtained data are encoded in an HSMM (II-A).

Reproduction of the encoded skill is achieved online. At each time-step a control command is obtained by solving an optimal control problem (II-B). The required objective function is constructed based on the information encoded in the HSMM as desecribed in Sections II-C and II-D.

A. Hidden Semi-Markov Model (HSMM)

The demonstrated data are encoded in an HSMM [10], [11], an extension of the Hidden Markov Model (HMM) in which the state[1] duration is explicitly modeled as a probability distribution. An HMM models a double stochastic process of which the observations are assumed to be generated by an underlying, unobservable, finite-state Markov chain. We represent each state by a single multivariate Gaussian $\mathcal{N}(\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)$, with

$$\boldsymbol{\mu}_i = \begin{bmatrix} \boldsymbol{\mu}_i^x \\ \boldsymbol{\mu}_i^{\dot{x}} \end{bmatrix}, \quad \boldsymbol{\Sigma}_i = \begin{bmatrix} \boldsymbol{\Sigma}_i^{xx} & \boldsymbol{\Sigma}_i^{x\dot{x}} \\ \boldsymbol{\Sigma}_i^{\dot{x}x} & \boldsymbol{\Sigma}_i^{\dot{x}\dot{x}} \end{bmatrix}, \tag{1}$$

modeling the local linear dynamics.

Variable duration modeling techniques, such as the HSMM, replace the self-transition probabilities $a_{i,i}$ of the HMM by an explicit model (non-parametric or parametric) of the relative time during which one stays in each state, see for example [11], [12]. Here, the duration will be modeled by a univariate Gaussian distribution $\mathcal{N}(\mu_i^{\mathcal{D}}, \Sigma_i^{\mathcal{D}})$.

The HSMM with K states is defined by the parameters $\{a_{i,j}, \Pi_i, \mu_i^{\mathcal{D}}, \Sigma_i^{\mathcal{D}}, \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i\}_{i,j}^K$, were Π_i are the initial state priors. These parameters are estimated using an Expectation Maximization (EM) algorithm [11], [13]. Various techniques exist to determine the number of states K, e.g. Bayesian information criterion [14], or Dirichlet processes [15].

B. Linear Unconstrained MPC

A Model Predictive Control (MPC) strategy is used to obtain a minimal intervention controller. MPC computes control commands based on system state predictions. This allows the controller to anticipate future events. MPC can be applied to a wide variety of systems, both linear and non-linear, with constraints on both input and output [16]. We use its simplest form: *linear unconstrained* MPC.[2]

The state predictions used in MPC are based on an estimate of the system model. We use a discrete linear system to describe the dynamic behavior of the system state

$$\boldsymbol{\xi}_{t+1} = \boldsymbol{A}\boldsymbol{\xi}_t + \boldsymbol{B}\boldsymbol{u}_t, \quad \boldsymbol{x}_t = \boldsymbol{C}\boldsymbol{\xi}_t, \tag{2}$$

with $\boldsymbol{A}$, $\boldsymbol{B}$ and $\boldsymbol{C}$ the system dynamics, input and output matrix, respectively. Based on this linear system, N_p state predictions $\boldsymbol{\xi}_r$, with $r \in \{t+1, \cdots, t+N_p\}$, are defined in terms of the current state $\boldsymbol{\xi}_t$, and N_c control commands

[1] Throughout this paper we explicitly distinguish *state*, to refer to the state of the HSMM, and *system state*, to refer to the state $\boldsymbol{\xi}$ of a dynamical system.

[2] Linear unconstrained MPC yields the same solution as finite horizon Linear Quadratic Regulator (LQR). In this work we refer to MPC rather than LQR to emphasize that control problem is solved at each time-step with a receding horizon.

$\boldsymbol{u}_r$ with $r \in \{t, \cdots, t+N_c-1\}$. The state predictions are compactly written in a matrix-vector form

$$\underbrace{\begin{bmatrix} \boldsymbol{A} \\ \boldsymbol{A}^2 \\ \boldsymbol{A}^3 \\ \vdots \\ \boldsymbol{A}^{N_p} \end{bmatrix}}_{\boldsymbol{S^\xi}} \boldsymbol{\xi}_t + \underbrace{\begin{bmatrix} \boldsymbol{B} & \boldsymbol{0} & \cdots & \boldsymbol{0} \\ \boldsymbol{AB} & \boldsymbol{B} & \cdots & \boldsymbol{0} \\ \boldsymbol{A}^2\boldsymbol{B} & \boldsymbol{AB} & \cdots & \boldsymbol{0} \\ \vdots & \vdots & \ddots & \boldsymbol{0} \\ \boldsymbol{A}^{N_p-1}\boldsymbol{B} & \boldsymbol{A}^{N_p-2}\boldsymbol{B} & \cdots & \boldsymbol{A}^{N_p-N_c}\boldsymbol{B} \end{bmatrix}}_{\boldsymbol{S^u}} \underbrace{\begin{bmatrix} \boldsymbol{u}_t \\ \boldsymbol{u}_{t+1} \\ \boldsymbol{u}_{t+2} \\ \vdots \\ \boldsymbol{u}_{t+N_c-1} \end{bmatrix}}_{\boldsymbol{U}}, \tag{3}$$

$$\boldsymbol{X} = \left(\boldsymbol{I}_{\boldsymbol{N_p}} \otimes \boldsymbol{C}\right)\left(\boldsymbol{S^\xi}\boldsymbol{\xi}_t + \boldsymbol{S}^u\boldsymbol{U}\right), \tag{4}$$

where $\boldsymbol{\zeta} = [\boldsymbol{\xi}_{t+1}^\top, \boldsymbol{\xi}_{t+2}^\top, \cdots, \boldsymbol{\xi}_{t+N_p}^\top]^\top$, $\boldsymbol{X} = [\boldsymbol{x}_{t+1}^\top, \boldsymbol{x}_{t+2}^\top, \cdots, \boldsymbol{x}_{t+N_p}^\top]^\top$, $\otimes$ is the Kronecker product and $\boldsymbol{I}_{N_p}$ is an $N_p \times N_p$ identity matrix.

The control command $\boldsymbol{u}_t$ is found by minimizing an objective function that defines the cost over the prediction horizon. We define a quadratic cost function

$$J = \sum_{r=t+1}^{t+N_p} \left(\hat{\boldsymbol{\xi}}_r - \boldsymbol{\xi}_r\right)^\top \boldsymbol{Q}_r \left(\hat{\boldsymbol{\xi}}_r - \boldsymbol{\xi}_r\right) + \sum_{r=t}^{t+N_c-1} \boldsymbol{u}_r^\top \boldsymbol{R}_r \, \boldsymbol{u}_r, \tag{5}$$

with $\boldsymbol{\xi_t}$ and $\hat{\boldsymbol{\xi}}_t$ representing the current and desired system state, respectively. The tracking cost $\boldsymbol{Q}_t$ influences the stiffness of the controller in the different elements of the system state. High tracking cost results in stiff control of the state variables. $\boldsymbol{R}_t$ is the control cost matrix. The control law that minimizes the cost J over time is optimal with respect to this objective. Rewriting the objective function in batch form, we obtain

$$J = \left(\hat{\boldsymbol{\zeta}} - \boldsymbol{\zeta}\right)^\top \boldsymbol{Q}\left(\hat{\boldsymbol{\zeta}} - \boldsymbol{\zeta}\right) + \boldsymbol{U}^\top \boldsymbol{R}\boldsymbol{U}, \tag{6}$$

with $\boldsymbol{Q} = \text{blockdiag}(\boldsymbol{Q}_{t+1}, \boldsymbol{Q}_{t+2}, \ldots, \boldsymbol{Q}_{t+N_p})$ and $\boldsymbol{R} = \text{blockdiag}(\boldsymbol{R}_t, \boldsymbol{R}_{t+2}, \ldots, \boldsymbol{R}_{t+N_c-1})$.

The vector of control commands $\boldsymbol{U}$ is obtained by substituting (3) into (6) and minimizing with respect to $\boldsymbol{U}$, yielding

$$\boldsymbol{U} = \left(\boldsymbol{S^{u\top}}\boldsymbol{Q}\boldsymbol{S^u} + \boldsymbol{R}\right)^{-1} \boldsymbol{S^{u\top}}\boldsymbol{Q}\left(\hat{\boldsymbol{\zeta}} - \boldsymbol{S^\xi}\boldsymbol{\xi}_t\right). \tag{7}$$

The computational cost of solving (7) heavily depends on the size of the inversion. This size can be reduced by selecting a control horizon $N_c < N_p$. An alternative, computationally more efficient, iterative solution for solving (5) subject to (2) exists. The presented batch solution, however, allows us to include, if required, additional constraints on both input and output forming a convex optimization problem [16].

C. State Sequence Synthesis

The construction of the objective function used in our approach is based on a state sequence $\boldsymbol{s} = \{s_1, \ldots, s_{N_p}\}$, with N_p the number of predictions, that is regenerated at each time step of the reproduction.

The generation process relies on the forward variable of the HSMM. The forward variable $\alpha_{i,t}$ defines the probability to be in state i at time step t given the observation $\{\boldsymbol{\xi}_1, \boldsymbol{\xi}_2, \ldots, \boldsymbol{\xi}_t\}$; i.e. $\mathcal{P}(i|\boldsymbol{\xi}_1, \boldsymbol{\xi}_2, \ldots, \boldsymbol{\xi}_t)$. It can be recursively computed with (see for example [13])

$$\alpha_{i,t} = \Pi_i \mathcal{N}_{t,i}^{\mathcal{D}} \prod_{r=1}^{t} \mathcal{N}_{r,i} + \sum_{j=1}^{K} \sum_{d=1}^{t-1} \alpha_{j,t-d} \, a_{j,i} \, \mathcal{N}_{d,i}^{\mathcal{D}} \prod_{r=t-d+1}^{t} \mathcal{N}_{r,i}, \tag{8}$$

when t is smaller than the time history d_{max}, otherwise

$$\alpha_{i,t} = \sum_{d=1}^{d^{max}} \sum_{j=1}^{K} \alpha_{j,t-d} \, a_{j,i} \, \mathcal{N}_{d,i}^{\mathcal{D}} \prod_{r=t-d+1}^{t} \mathcal{N}_{r,i}, \tag{9}$$

with $\mathcal{N}_{r,i} = \mathcal{N}\left(\boldsymbol{\xi}_r | \, \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i\right)$ and $\mathcal{N}_{d,i}^{\mathcal{D}} = \mathcal{N}(d|\mu_i^{\mathcal{D}}, \Sigma_i^{\mathcal{D}})$.

At each time step the forward variable is used for two purposes. First, to keep track of the probability $\mathcal{P}(i|\boldsymbol{\xi}_1, \boldsymbol{\xi}_2, \ldots, \boldsymbol{\xi}_t)$. Here, $\alpha_{i,t}$ is computed while taking into account the current system state $\boldsymbol{\xi}_t$. This process is initialized with the priors, i.e. $\alpha_{i,0} = \Pi_i$. Second, N_p predictions are computed to create the state sequence prediction $\boldsymbol{s}$ with

$$s_r = \underset{i \in \{1, \cdots, K\}}{\arg\max} \; \alpha_{i,r}, \; \forall \, r \in \{t+1, t+2, \cdots, t+N_p\}. \tag{10}$$

When computing the predictions, we assume equal observation probability for all states, i.e. $\mathcal{N}_{r,i} = 1 \; \forall i$.

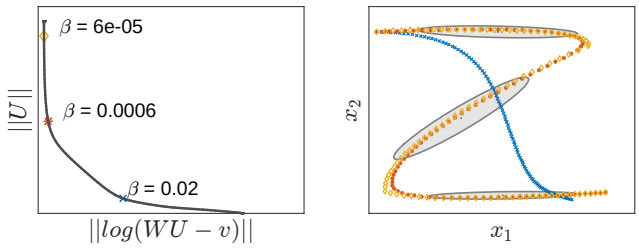
The combination of observed system state, through $\mathcal{N}_{r,i}$, and state duration, through $\mathcal{N}_{d,i}^{\mathcal{D}}$, enables the movement reproduction with variable duration. The observations influence the probability of the system to be in a certain state, and can alter the movement through the state sequence evolution. The movement is slowed down by observations preventing state transition; i.e. observations that are likely in the current or previous states. It is sped up by observations that encourage state transition; i.e. observations that are more likely to occur in the future states. The influence that observations have in the state evolution is regulated by $\mathcal{N}_{d,i}^{\mathcal{D}}$, specifying the mean and variance of the state duration. If the duration variance is small, the influence of the observations will diminish, removing temporal variability. If it is large, the state evolution will be governed by the observations.

D. Setting the Objective

The control behavior is shaped by $\hat{\boldsymbol{\xi}}_r$, $\boldsymbol{Q}_r$ and $\boldsymbol{R}_r$. To create a minimial intervention controller, we assume that low-variability in the demonstration data indicates an area where accuracy is required, and high variability indicates an area where less accuracy is required similarly to our previous work [8].

In this work, however, we do not encode the joint probability density function $\mathcal{P}(\boldsymbol{\xi}, t)$. Instead, we use an HSMM encoding $\mathcal{P}(\boldsymbol{\xi})$ in combination with a duration model that replaces the explicit temporal signal. The reference $\hat{\boldsymbol{\zeta}}$, previously obtained through Gaussian Mixture Regression (GMR), is replaced by a trajectory based on the state sequence $\boldsymbol{s} = \{s_1, \ldots, s_{N_p}\}$ that is synthesized as described in Section II-C. The result is a piecewise reference consisting of the centers and covariance of the Gaussian kernels, i.e.

$$\hat{\boldsymbol{\xi}}_r = \boldsymbol{\mu}_{s_r}, \qquad \boldsymbol{Q}_r = (\boldsymbol{\Sigma}_{s_r})^{-1}. \tag{11}$$



(a) L-curve (b) Resulting trajectory

Fig. 2: Visualization of the effect of the regularization parameter β. *Left:* Three different β values are selected on the curve visualizing the trade-off between reproduction accuracy $||\boldsymbol{WU} - \boldsymbol{v}||$ and control effort $||\boldsymbol{U}||$. *Right:* The colored dots indicate the different reproductions for different values of β, and the gray ellipsoids represent one standard deviation of the Gaussian kernels encoding the local movement dynamics. As β decreases the smoothness of the motion increases, but the tracking accuracy decreases.

Figure 1 provides a 1D-example of the reproduction using the piecewise reference. The optimization, performed by MPC, smooths the piecewise reference by creating a trade-off between the desired position and velocity defined in the system state $\hat{\boldsymbol{\xi}}_r$. The weights of this trade-off are defined by the precision matrices $\boldsymbol{Q}_r$ encoding the full covariance between the system state variables.

The optimization requires a minimal number of state-predictions for the generation of smooth trajectories. These are obtained through state sequences $\boldsymbol{s}$ that cover multiple states. Too short state sequences, i.e. shorter than the duration of a single state, will lead to jumps between local minima created by each state. We empirically found that using twice the average state duration, $N_p > \frac{2}{K}\sum_{i=1}^{K}\mu_i^{\mathcal{D}}$, results in proper reproductions.

Equation (7), is the solution to a regularized least-squares problem known as *Tikhonov regularization* or *ridge regression* [17]. This can be seen by reformulating the minimization (6) subject to (3) into the regularized least-squares objective

$$\min_{\boldsymbol{U}} \Big(|| \underbrace{\boldsymbol{L}\boldsymbol{S}^{\boldsymbol{u}}}_{\boldsymbol{W}} \boldsymbol{U} - \underbrace{\boldsymbol{L}(\boldsymbol{\zeta} - \boldsymbol{S}^{\boldsymbol{\xi}}\boldsymbol{\xi}_t)}_{\boldsymbol{v}} ||_2^2 + \beta ||\hat{\boldsymbol{U}}||_2^2 \Big), \qquad (12)$$

where $\boldsymbol{L}$ is obtained through the Cholesky decomposition $\boldsymbol{Q} = \boldsymbol{L}^\top \boldsymbol{L}$, and we have assumed $\boldsymbol{R}_r = \beta^2 \boldsymbol{I}_D$.

β controls the trade-off between model accuracy $||\boldsymbol{WU} - \boldsymbol{v}||_2^2$ and large values of $\boldsymbol{U}$. This trade-off generally has a sharp L-like shape [18], where β can be estimated as the first point with significant slope change. Figure 2 shows the relation between the L-curve and the resulting least-squares solution. This insight can be used to select the control cost.

E. Task-Parameterized Extension

In [8], we introduced a Task-Parameterized GMM (TP-GMM). This type of encoding enables generalization of the movement to situations with new task parameters.

Instead of encoding the demonstration data directly, it is first projected into P local frames of reference using linear transformations defined by the task parameters $\boldsymbol{b}^{(p)}$ and $\boldsymbol{A}^{(p)}$.[3] The extended dataset is clustered into a HSMM with K states $\mathcal{N}(\boldsymbol{\mu}_i^{(p)}, \boldsymbol{\Sigma}_i^{(p)})$ in P coordinate systems.

During reproduction the states are projected from their local frames into the global frame of reference using the current task parameters. There the P states i are combined into one using the product of Gaussian $\forall i \in 1, \dots, K$. The resulting K states are used to evaluate the forward variable (8), and to construct the objective function (6) as described in Sections II-C and II-D, respectively. This process is repeated at each time step to allow for online changing task parameters.

III. COMPARISON EXPERIMENT

Figure 3 shows reproductions of a minimal intervention controller using three different cost functions. In tGMM, the cost function is based on a GMM with explicit time encoding as described in [8]. It is compared to two HSMM approaches that are distinguished by online and offline synthesis of the piecewise reference.

Figure 3a shows the reproductions in the unperturbed case. All reproductions lie within the area of the demonstrated trajectories, and their control inputs have the same order of magnitude. This was achieved by setting the cost factor β to 0.001 and 0.007 for tGMM and the HSMM reproductions, respectively. These settings provide a proper baseline to compare the methods in case of perturbations.

The mismatch between online and offline HSMM is due to the observation probabilities, these cause a difference between the online and offline generated state sequence.

Figure 3b shows the reproductions of the systems under perturbation. The systems are perturbed by fixing the its state $\boldsymbol{\xi}$ over the time interval $[0.4, 0.6]$. This is visualized on the temporal plots in the upper right corner of the figure.

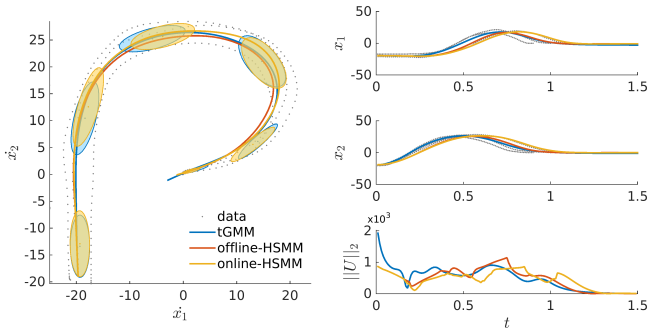
There is a clear difference between the online and offline HSMM. The online HSMM is able to spatially reproduce the encoded motion, but requires more time for the reproduction. The offline method reaches the final state within the demonstrated time window, but is not able to match the demonstration spatially. Offline and online state sequence synthesis is the same when the duration model has a variance that approaches zero for all states. The difference between these two reproductions demonstrates the potential of HSMM to encode different responses to temporal perturbation.

The time-based GMM is able to reproduce the encoded motion with spatial and temporal accuracy. There is, however, a clear difference in magnitude of the control commands (shown in the bottom right plot of Figure 3). The time-based GMM shows much higher interaction forces compared to HSMM-based reproductions.

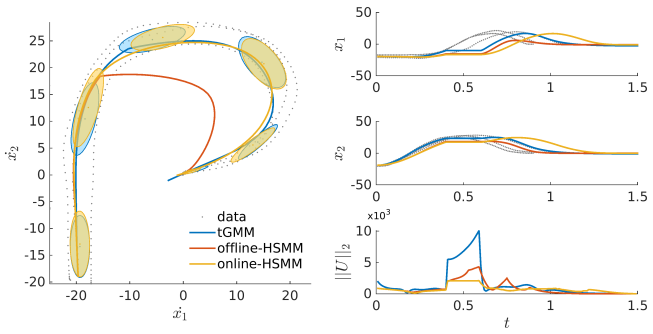
IV. ROBOT EXPERIMENT

The proposed method is evaluated in an object picking scenario. The objective is to teach the robot to pick up an

[3] $\boldsymbol{A}^{(p)}$ is used to comply with the original notation used in TP-GMM, it should not be confused with the system dynamics $\boldsymbol{A}$ that is defined in Section II-B.



(a) Unperturbed reproductions



(b) Perturbed reproductions

Fig. 3: Reproduction attempts of a minimal intervention controller based on three different strategies; (i) time-based GMM (tGMM), (ii) piecewise reference based on pre-determined state sequence (offline-HSMM), (iii) piecewise reference synthesized online (online-HSMM). The colored ellipsoids represent one standard deviation of the Gaussian kernels encoding the local movement dynamics. Their colors correspond to the model (blue: tGMM, red and yellow: HSMM). See Section III for details.

object from different locations. This scenario allows us to demonstrate the capabilities of the proposed method, namely the ability to encode movements, to react to changes in the environment, and to respond to spatial and temporal perturbations.

The experiment is performed using a Barrett WAM, a torque controlled robot with 4 degrees of freedom. We exclude the grasping problem from the task by using adhesive surfaces to pick up objects. The object is placed on a marker which is tracked by an OptiTrack system.

The experiment consists of a demonstration phase and a reproduction phase. During the demonstration phase kinesthetic teaching is used to show the robot how to move from a start position towards the object, and back. Demonstrations are given for different locations of the marker. During the demonstration the marker is not moved.

We use the TP-HSMM to handle the varying position of the object. We pre-define two frames of reference in this experiment ($P = 2$). The task parameters $\{\boldsymbol{b}^{(1)}, \boldsymbol{A}^{(1)}\}$ and $\{\boldsymbol{b}^{(2)}, \boldsymbol{A}^{(2)}\}$ represent the origin of the WAM and the marker, respectively. The orientation of the marker is set to $\boldsymbol{A}^{(2)} = \boldsymbol{I}$ in this experiment.

During the demonstrations the positions of the end-effector and the marker are recorded with a sampling frequency of 100 Hz. Each demonstration results in a dataset $\mathcal{X}_n \in \mathbb{R}^{2D \times P \times T_n}$, a tensor representing the Cartesian ($D = 3$) position $\boldsymbol{x}$ and velocity $\dot{\boldsymbol{x}}$ of the end-effector in the WAM and the marker frame for T_n time steps. The velocity is obtained by a first-order Euler approximation ($\dot{\boldsymbol{x}}_t = \frac{\boldsymbol{x}_t - \boldsymbol{x}_{t-1}}{\Delta t}$).

During reproduction the robot is torque controlled. The torque command

$$\boldsymbol{\tau} = \boldsymbol{\tau}_g + \boldsymbol{J}^\top(\boldsymbol{q})\boldsymbol{u} \tag{13}$$

consists of gravity compensation term $\boldsymbol{\tau}_g$, and the MPC control command $\boldsymbol{u}$. The control command is transformed to joint space using the Jacobian $\boldsymbol{J}(\boldsymbol{q})$ that depends on the joint angles $\boldsymbol{q}$.

The linear MPC model of the end-effector is defined by

$$\boldsymbol{A} = \begin{bmatrix} \boldsymbol{0} & \boldsymbol{I}_3 \\ \boldsymbol{0} & \boldsymbol{0} \end{bmatrix} \quad \boldsymbol{B} = \begin{bmatrix} \boldsymbol{0} \\ \boldsymbol{I}_3 \end{bmatrix}. \tag{14}$$

These settings assume an end-effector with unit mass, and neglects the friction and the inertia of the robot. The prediction horizon is set to $N_p = 100$ and the control horizon is set to $N_c = 30$. The MPC reproduction loop runs at $100Hz$.

A. Results

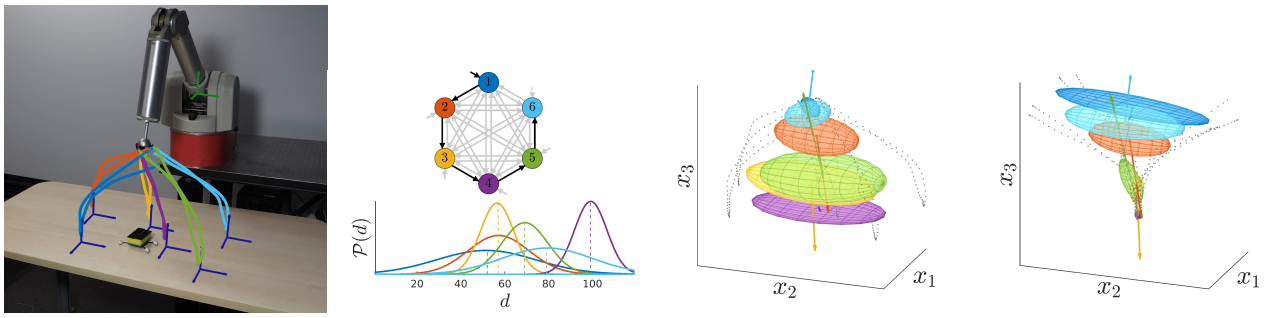
Six demonstrations of the skill are given for different locations of the object (see Figure 4a).

The demonstration data are used to train a TP-HSMM with $K = 6$ states (empirically set) using EM. The parameters of the HSMM that encode the local dynamics ($\boldsymbol{\mu_i}$ and $\boldsymbol{\Sigma}_i$) are initialized by dividing the data of each demonstration into K equal parts and calculating their corresponding means and covariances. The transition was initialized with parameters corresponding to a left-right model.

The resulting TP-HSMM after EM is displayed in Figure 4. The duration model shows that the initial and the final state have the largest variance. This is caused by the inconsistent starting and stopping while recording demonstrations. The movement towards and away from the marker shows a smaller variance. The duration of the pick-up phase, the purple state #4, is relatively long. Here the demonstrator pushes the object to make sure that it adheres to the end-effector. The covariances of the Gaussians in the different frames of reference show that the desired position at the start and end position are governed by the WAM frame, while the pick-up location is governed by the location of the marker.

Three reproduction conditions were considered to show that the proposed approach can cope with temporal disturbance; (i) Unperturbed: the robot executes the movement without external perturbations; (ii) Hold back: the movement is slowed down by holding the robot; (iii) Pushed: the movement is sped up by moving the robot above the motion's nominal speed.

The results of the reproductions are displayed in Figures 5 and 6. Figure 5 shows that the system state $\boldsymbol{\xi}$ does not always converge to the centers of the Gaussians. Instead, the system state converges to the trade-off between position and velocity. The weight of this trade-off is given by the inverse of the covariance. The control input in Figure 5 also shows



(a) Demonstrations (b) Duration Model (c) Model: WAM Frame (d) Model: Marker Frame

Fig. 4: Visualization of the encoded TP-HSMM. (a) The experimental setup and the provided demonstrations. (b) States transition model with their corresponding duration. (c) and (d) display the demonstration data (in gray), and the location of the Gaussian kernels in the two frames of reference. The colored ellipsoids represent the Gaussian kernels. Their colors correspond to the states of the HSMM. The arrow that originates from the center of a Gaussian indicates the mean of the encoded velocity.

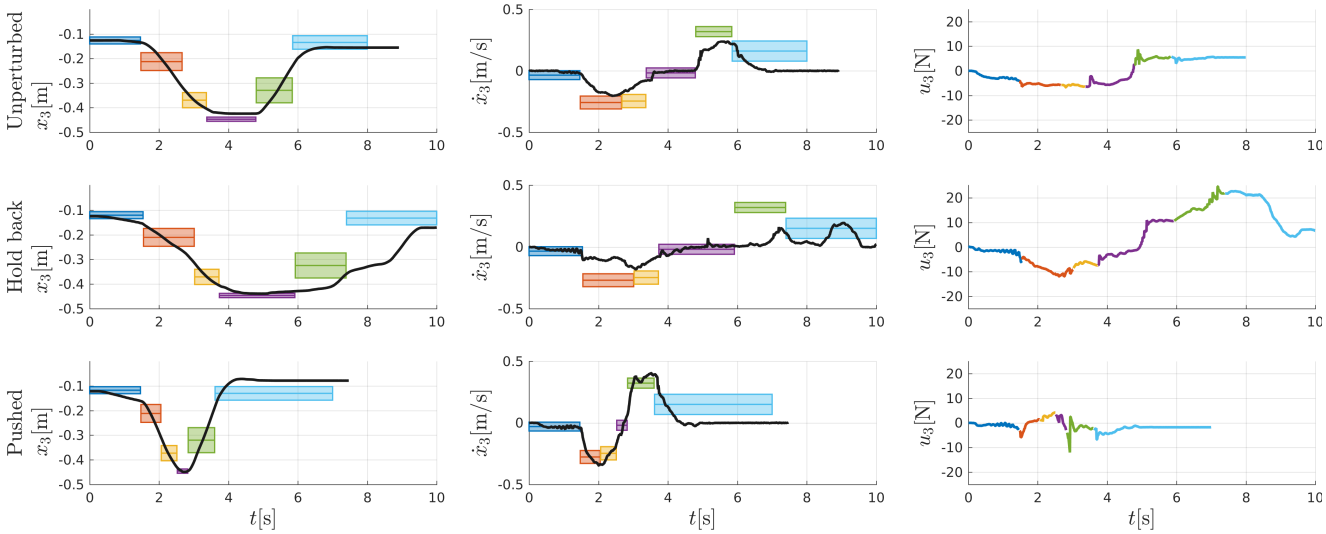


Fig. 5: The position x_3, velocity $\dot{x}_3$, and control input u_3 over time obtained during the reproduction of the pick up task under three different conditions. The different rows correspond, top to bottom, to the unperturbed, holding, and pushing conditions. The blocks in the position and velocity graphs indicate the most likely state at each time step. The control input has been colored to emphasize the most likely state at each time step. The colors in the graph correspond to the Gaussians in Figure 4b.

how much resistance the robot gave against the different perturbations.

Figure 7 shows how the robot is able to adapt its motion to a changing pick-up position. Initially the robot moves towards the object positioned at the far right. When moving the object, the robot continuously adapts its movement plan, and is able to successfully pick up the object.

V. DISCUSSION

Besides the added temporal variability, the proposed approach offers a number of advantages for PbD over existing non-autonomous methods such as DMP [5], DS-GMR [8] and ProMP [4]. It adds generality since the Markov chain structure of HSMM allows to encode both cyclic and non-cyclic behavior. In addition, the lack of the explicit temporal signal reduces the need for temporal realignment, and allows learning from partial demonstrations without additional modifications.

A difficulty we encountered with the proposed model is scarcity of the data available to estimate the duration model of the HSMM. Given that each transition, for discrete movements, is only visited once per demonstration, each demonstration represents only one datapoint to train the duration model. As a result, outliers have a large effect on the variance and the mean of the duration. This can be toned down by introducing a prior in the form of a minimum variation. The encoding of position and velocity does not suffer from this drawback because each demonstration provides several datapoints to estimate them.

The proposed method has links with trajectory GMM, a method used in speech synthesis [19], [20], [12]. Similarly to our approach, dynamical features are approximated by local linear models. Synthesis of a trajectory $\boldsymbol{X}$ is achieved by maximizing the likelihood of the dynamics $\boldsymbol{\zeta} = [\boldsymbol{\xi}_1^\top, \dots, \boldsymbol{\xi}_T^\top]^\top$ given a state sequence $\boldsymbol{s}$. Trajectory GMM enforces a smooth trajectory using the constraint $\boldsymbol{\zeta} = \boldsymbol{\Phi}\boldsymbol{X}$. Here, $\boldsymbol{\Phi}$ defines the relation between the trajectory $\boldsymbol{X}$ and its dynamical

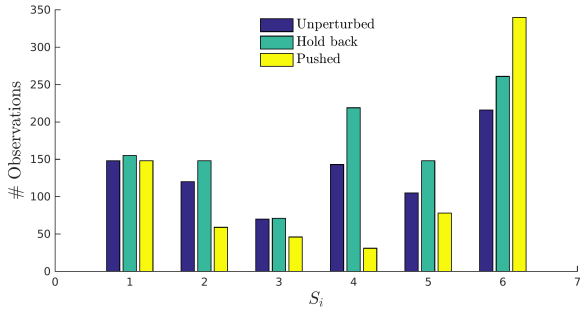


Fig. 6: Visualization of the state duration for reproductions under three different conditions. Each bar indicates the number of observations made in that state for the corresponding reproduction. The state of an observation is given by $i = \underset{i \in 1,\ldots,K}{\arg\max}\,(\alpha_{i,t})$.

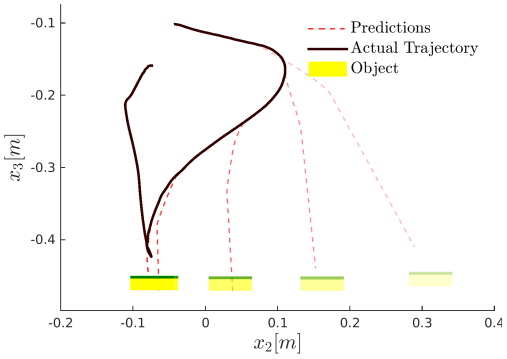


Fig. 7: Visualization of the ability to adapt to changing object location online. The figure shows how the predicted path changes when the object is moved. The decrease in transparency indicates the motion of the object over time.

features $\boldsymbol{\zeta}$. Instead, we use the dynamical system (2) as a constraint to ensure this smoothness. The use of the dynamical system constraint allows us to directly compute the control commands $\boldsymbol{U}$, and introduce a cost on control effort yielding a minimal intervention controller.

VI. CONCLUSION & FUTURE WORK

We presented a PbD approach to learn a minimal intervention controller. The novelty of our approach is the application of the minimal intervention principle to the movement duration. Here we have assumed that we can relate the task performance in terms of duration to the temporal variation observed during demonstration.

The implementation of the controller is based on the combination of MPC with an HSMM encoding the demonstrated skill. We have demonstrated that MPC properly smooths the piecewise objective function generated from HSMM. This, combined with the online synthesis of the reference, creates an interactive controller that responds adequately to both spatial and temporal perturbations.

We compared the reproduction results of our approach with an approach that does not take temporal variability into account. The simulation showed that our approach is able to respond to perturbations with lower interaction forces, and, depending on the duration model, favors either temporal or spatial task performance. Finally, we have demonstrated our method in a robotic experiment, showing the ability to adapt to changes in the task online.

The MPC formulation used in this work allows us to include additional (convex) constraints on both input and output. We plan to exploit such constraints in future work to improve safety in human-robot interaction.

The current implementation controls the robot in task space. In future work, we plan to extend our method in such a way that task objectives are defined in task space while considering the controller in joint space.

REFERENCES

[1] B. D. Argall, S. Chernova, M. Veloso, and B. Browning. A survey of robot learning from demonstration. *Robot. Auton. Syst.*, 57(5):469–483, 2009.

[2] S.M. Khansari-Zadeh and A. Billard. Learning control lyapunov function to ensure stability of dynamical system-based robot reaching motions. *Robot. Auton. Syst.*, 62:752–765, 2014.

[3] S. Calinon, A. Pistillo, and D. G. Caldwell. Encoding the time and space constraints of a task in explicit-duration hidden Markov model. In *Proc. IEEE/RSJ IROS*, pages 3413–3418, San Francisco, CA, USA, September 2011.

[4] A. Paraschos, C. Daniel, J. Peters, and G. Neumann. Probabilistic movement primitives. In *Proc. NIPS*, pages 2616–2624. Curran Associates, Inc., 2013.

[5] A. Ijspeert, J. Nakanishi, P. Pastor, H. Hoffmann, and S. Schaal. Dynamical movement primitives: Learning attractor models for motor behaviors. *Neural Computation*, 25(2):328–373, 2013.

[6] S. Calinon, Z. Li, T. Alizadeh, N. G. Tsagarakis, and D. G. Caldwell. Statistical dynamical systems for skills acquisition in humanoids. In *Proc. IEEE Humanoids*, pages 323–329, Osaka, Japan, 2012.

[7] E. Todorov and M. I. Jordan. Optimal feedback control as a theory of motor coordination. *Nature Neuroscience*, 5:1226–1235, 2002.

[8] S. Calinon, D. Bruno, and D. G. Caldwell. A task-parameterized probabilistic model with minimal intervention control. In *Proc. IEEE ICRA*, pages 3339–3344, Hong Kong, China, May-June 2014.

[9] J. R. Medina, D. Lee, and S. Hirche. Risk-sensitive optimal feedback control for haptic assistance. In *Proc. IEEE ICRA*, pages 1025–1031, May 2012.

[10] S. E. Levinson. Continuously variable duration hidden Markov models for automatic speech recognition. *Computer Speech & Language*, 1(1):29–45, 1986.

[11] S.-Z. Yu. Hidden semi-Markov models. *Artificial Intelligence*, 174:215–243, 2010.

[12] H. Zen, K. Tokuda, and T. Kitamura. Reformulating the HMM as a trajectory model by imposing explicit relationships between static and dynamic feature vector sequences. *Computer Speech and Language*, 21(1):153–173, 2007.

[13] L. R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proc. IEEE*, 77:2:257–285, February 1989.

[14] G. Schwarz. Estimating the dimension of a model. *Annals of Statistics*, 6(2):461–464, 1978.

[15] S. P. Chatzis, D. Korkinof, and Y. Demiris. A nonparametric Bayesian approach toward robot learning by demonstration. *Robotics and Autonomous Systems*, 60(6):789–802, 2012.

[16] F. Borrelli, A. Bemporad, and M. Morari. *Predictive Control for linear and hybrid systems*. 2015. In preparation.

[17] R. C. Aster, B. Borchers, and C. H. Thurber. *Parameter estimation and inverse problems*. Academic Press, 2013.

[18] P. C. Hansen. *The L-curve and its use in the numerical treatment of inverse problems*. IMM, Dep. of Math. Modelling, Tech. Univ. of Denmark, 1999.

[19] S. Furui. Speaker-independent isolated word recognition using dynamic features of speech spectrum. *IEEE Trans. on Acoustics, Speech, and Signal Processing*, 34(1):52–59, 1986.

[20] K. Sugiura, N. Iwahashi, H. Kashioka, and S. Nakamura. Learning, generation, and recognition of motions by reference-point-dependent probabilistic models. *Advanced Robotics*, 25(6-7):825–848, 2011.