Online Incremental Learning of Manipulation Tasks for Semi-Autonomous Teleoperation

Ioannis Havoutis[1], Ajay Kumar Tanwani[1,2], Sylvain Calinon[1]

***Abstract*— We present an approach for online incremental learning of manipulation tasks. A Bayesian clustering algorithm is used to build an online *hidden semi-Markov model* (HSMM) that captures state transition and state duration probabilities of demonstrated motions. Motions are then generated by stochastically sampling from the learned model and tracked using an infinite-horizon *linear quadratic regulator* (LQR), with varying stiffness and damping characteristics learned from the demonstrations. Our approach provides a compact skill representation that is learned online and can be used in a semi-autonomous teleoperation setting, where direct teleoperation is infeasible due to large communication latency or failure. We present a planar drawing example to highlight the flexibility of our approach, and demonstrate how our solution can be used for a *hot-stabbing* motion in an underwater teleoperation scenario. We evaluate the performance of our approach over multiple trials and report high accuracy and repeatability, resulting to consistently successful hot-stabbing motions.**

I. Introduction

Many useful robotics applications require performing tasks in environments that are not friendly for humans. One typical example is underwater activities, ranging from inspection and maintenance of underwater cables and pipelines, to underwater archaeology and marine biology. To this end there has been a boom in underwater *remotely operated vehicles* (ROVs) over the past few years. Nonetheless the cost of using ROVs is still prohibitively high for wider adoption, as currently ROV usage still requires substantial off-shore support.

One of the main limiting factors is that a large off-shore crew is required to supervise and teleoperate the ROV directly from the support vessel. This is mainly due to the need of online teleoperation, i.e. the operator receives visual feedback from an array of cameras on the ROV and accordingly uses a set of buttons, knobs and joysticks to guide the motion of all, body and arm(s), degrees-of-freedom (DoF) of the ROV. This cost can be reduced by moving the support and teleoperation team to an on-shore facility and communicating with the ROV remotely. Current satellite communications technology suffers from large latencies and deems traditional *direct* ROV teleoperation infeasible.

We are developing a novel teleoperation paradigm within which no direct teleoperation of the ROVs' DOFs is required but control is locally handled (onboard) using a probabilistic representation of task/skill primitives. Such a representation can adapt to changing task parameters and is robust against intermittent communication. Within the DexROV project [1], we are investigating an efficient encoding of manipulation tasks that is learned via programming by demonstrations.

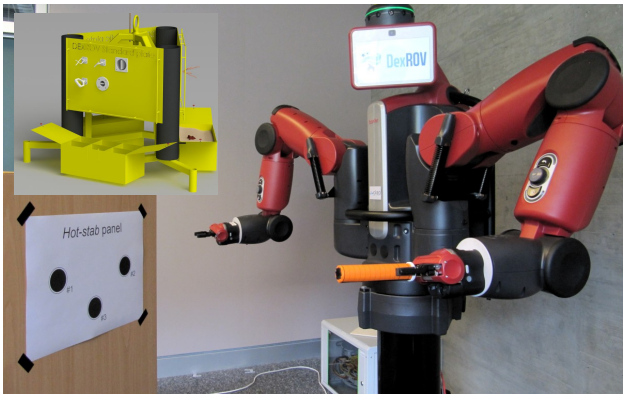


Fig. 1. The Baxter robot being taught how to perform *hot-stabbing* motions. The orange cylinder is a mock-up of the hot-stab plug while on the left a mock-up of a hot-stab panel with three receptacles is set up. The inset image shows a rendering of the underwater evaluation panel used in DexROV, housing 3 plugged-in hot-stabs (with different handles) and 2 rotational switches.

In our previous work [2], we proposed the use of a task-parameterized semi-tied hidden semi-Markov model (HSMM) for robustly learning robot manipulation tasks in a batch manner. In this paper, we are interested in *incrementally* building such model from demonstrated motions using an online DP-Means algorithm [3], that generates motion plans by stochastically sampling from the learned model. The generated motion is tracked by an infinite-horizon linear quadratic regulator (LQR) that yields smooth trajectories with varying stiffness/compliance characteristics learned from the demonstrations.

The main contribution of this work is the online and incremental learning of HSMM motion representations that can be used to encode manipulation tasks within a semi-autonomous teleoperation scenario. We show how our system can be used with a failing communication example and evaluate the performance of a learned ROV task, reporting averaged results over multiple trials. With our approach we are able to add datapoints incrementally, without the need to re-train the model in a batch fashion, and by discarding the demonstration datapoints after observation. We show how such skills can be learned and how this model can decouple the local control from the teleoperation setup. In fact, with our approach only a small set of model parameters needs to

[1]Idiap Research Institute, Martigny, Switzerland. {ioannis.havoutis, ajay.tanwani, sylvain.calinon}@idiap.ch

[2]Ecole Polytechnique Federale de Lausanne (EPFL), Switzerland.

This work was in part supported by the DexROV project through the EC Horizon 2020 programme (Grant #635491).

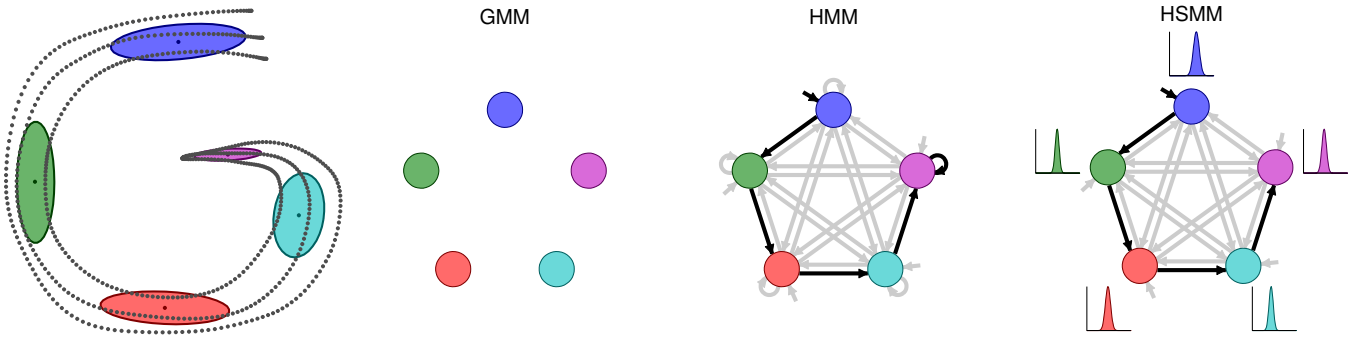


Fig. 2. Differences of transition representations in GMM, HMM and HSMM. The left graph shows three demonstrations represented as black dots, and the Gaussian clusters (states) as colored ellipses (isocontours of one standard deviation). A GMM model encodes the structure of the motion but does not model the transition between the states. An HMM uses transition and self-transition probabilities to model the state connectivity. Self-state transitions are known to only poorly describe the probability that the system is expected to stay in one of the states. The HSMM model instead explicitly models the state duration probabilities as Gaussian distributions, while keeping the transition probabilities between states.

be communicated from the operator side to the teleoperated system. This makes the overall method robust to intermittent communication and large latency.

The rest of the paper is structured as follows. In Section II we discuss relevant and previous work. Section III describes our approach of extending a *Gaussian mixture model* (GMM) to an HSMM. We then present a planar drawing motion example in Section IV, that highlights the advantages of the proposed method. Section V presents our experimental setup and Section VI describes our experimental trials for teaching a Baxter robot to perform *hot-stabbing* motions, a common routine task for underwater ROVs. In Section VI-A we evaluate the performance of the learned hot-stabbing task, reporting results averaged over multiple trials. Finally we conclude our work with a discussion on our results followed by an overview of future research directions.

II. Related work

A popular *learning by demonstration* (LbD) approach is to use *Dynamic Movement Primitives* (DMPs) [4]. DMPs are dynamical systems with simple convergence behaviour that are coupled with a learned non-linear function that modulates their output. This way, DMPs can provide adaptive motion representations that are easy to implement. One drawback of standard DMPs is that a sequence of radial basis activation functions needs to be allocated manually (usually spread equally in time with a shared variance), and that each DoF of the system is separately described (synchronized by a phase variable), sometimes leading to sets of DMPs that have difficulty in capturing joint synergies if too few basis functions are used.

An alternative LbD approach is to encode a motion in a GMM and use Gaussian Mixture Regression (GMR) to regenerate the motion. It was shown in [5] that a DMP can be recast as a GMR problem (for a GMM with diagonal covariance), and that its natural extension to a GMM with full covariances can represent local coordination patterns. With this probabilistic form of DMP, the basis functions can also be learned from the demonstrations.

Hidden Markov Models (HMMs) have been used in robotics in a number of approaches. For example Lee and Ott proposed to combine HMM with GMR to cope with the poor duration modeling properties of standard HMMs [6], [7]. Similarly, Chan *et al.* used GMR and HMM as part of a constrained manipulator visual servoing controller [8]. Bowen and Alterovitz presented a combination of an HMM for task representation and a sampling-based motion planner to produce (asymptotically) optimal plans [9]. Kulic *et al.* used HMMs to incrementally group together human whole-body motions, using hierarchical agglomerative clustering, based on their relative distance in HMM space [10].

Often, the use of HMM-based approaches in robotics applications is limited by the simplistic state duration modeling that HMMs provide. Other signal processing related disciplines, such as speech synthesis, have developed a number of models that seek to model state duration information more explicitly (for an overview see [11]). One such model is the Hidden Semi-Markov Models (HSMM) [12]. Recently we experimented with the use of HSMM in robot applications, by contrasting it to a set of different graphical model based approaches [13]. HSMMs are relevant for robot motion generation because they model the transitions and the durations of state activations, thus providing a relative time instead of a global time representation. In [2], we exploited this local duration representation for autonomously learning and reproducing in new situations the challenging manipulations tasks of opening a valve and moving objects while avoiding obstacles.

The approach that we propose in this paper for online HSMM estimation draws parallels to the DP-means extension to HMM presented in [14]. There the authors present a small-variance asymptotic analysis of the HMM and its infinite-state Bayesian nonparametric extension. Our solution is based on simpler assumptions that nonetheless work well in practice for the underwater teleoperation application that we consider.

III. Approach

We developed a method that leverages the advantages of an online GMM building algorithm, namely the DP-means

Algorithm 1 *Online hidden semi-Markov model learning*

Input: $< \lambda, \gamma >$

procedure Online HSMM

Initialize $K := 1, N_k := 1, \{d, c_{k,k}, \mu_k^{D^{(old)}}, e := 0\}$
while new $\boldsymbol{\xi}_t$ is being added **do**
  Compute $d_{t,i} = \|\boldsymbol{\xi}_t - \boldsymbol{\mu}_i\|_2 \qquad i = 1 \dots K$
  if $\min_i d_{t,i} > \lambda$ **then**
    $K := K + 1, \quad \pi_k := \frac{1}{t}, \quad \boldsymbol{\mu}_k := \boldsymbol{\xi}_t, \quad \boldsymbol{\Sigma}_k := \gamma \boldsymbol{I}$
  else
    $q_t = \operatorname{argmin}_i d_{t,i}$
    Update $\pi_{q_t}, \boldsymbol{\mu}_{q_t}, \boldsymbol{\Sigma}_{q_t}$ as described in [15]
  end if
  if $q_t = q_{t-1}$ **then** # $(t > 1)$
    $d := d + 1$
  else
    $c_{q_{t-1},q_t} := c_{q_{t-1},q_t} + 1$
    $a_{q_{t-1},q_t} := c_{q_{t-1},q_t} / \sum_{k=1}^{K} c_{q_{t-1},k}$
    $\mu_{q_{t-1}}^D := \mu_{q_{t-1}}^{D^{(old)}} + \frac{(d - \mu_{q_{t-1}}^{D^{(old)}})}{N_{q_t}}$
    $e := e + (d - \mu_{q_{t-1}}^{D^{(old)}})(d - \mu_{q_{t-1}}^D)$
    $\Sigma_{q_{t-1}}^D := \frac{e}{(N_{q_t} - 1)}$ # $(N_{q_t} > 1)$
    $d := 0, \quad N_{q_t} := N_{q_t} + 1, \quad \mu_{q_{t-1}}^{D^{(old)}} := \mu_{q_{t-1}}^D$
  end if
end while
return $\theta^* = \{\pi_i, \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i, \{a_{i,j}\}_{j=1}^K, \mu_i^D, \Sigma_i^D\}_{i=1}^K$

clustering [3]. This uses insights from Bayesian nonparametric approaches to arrive to a hard clustering approach that is online and incrementally built, and scales to large datasets.

In the interest of space, we only provide a brief overview of DP-means clustering. The interested reader can refer to [3] for further details. A GMM is incrementally built by incorporating datapoints $\boldsymbol{\xi}_t \in \mathbb{R}^D$, where D is the dimensionality of the problem at hand. For each new datapoint, the squared Euclidean distance to the GMM cluster centers is calculated. If this distance is greater than a threshold based on the size/range of the motion, a new cluster is added to the GMM. If the distance to the nearest cluster is lower than the threshold, then the cluster components, $\boldsymbol{\mu}_i$ and $\boldsymbol{\Sigma}_i$ (and cluster prior π_i), are updated according to the MAP estimate described in [15]. This results in a model that is incrementally expanded with more clusters (Gaussians) if the need arises, can be incrementally built – for example in a number of demonstrations – and is built online, i.e. no batch processing step is needed, while no data is stored. This makes the approach particularly appealing for large and incrementally growing sets of demonstrations.

A. Extension to HMM

Overall a GMM is built where each cluster, q_i with $i = 1 \dots K$, is described by a single Gaussian distribution $\mathcal{N}(\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)$. In the context of HMM, each cluster (or node) is a Gaussian component q_i in the model, that encodes the evolution of a set of variables $\boldsymbol{\xi}_t$.

An HMM considers transition probabilities between the K Gaussians, that form a $K \times K$ transition probability matrix, where each element $a_{i,j}$ in the matrix represents the probability to move to the state q_j, while currently being in state q_i, and Π_i are initial emission probabilities. The parameters of an HMM are described by $\boldsymbol{\Theta} = \{\{a_{i,j}\}_{j=1}^K, \Pi_i, \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i\}_{i=1}^K$, and can be estimated with an EM algorithm, see for example [11], in case data is presented in batches. Here, we instead optimize the parameters *online* from the DP-means estimate.

Each time a new demonstration is presented, for each datapoint that is added to the GMM, we estimate to which Gaussian component it most likely belongs. This way, for each datapoint $\boldsymbol{\xi}_t$ we can estimate the state q_j and the previous state q_i. To build up the transition probabilities, $a_{i,j}$, we keep a matrix $c_{i,j}$, $\boldsymbol{c} \in \mathbb{R}^{K \times K}$, that counts the number of state transitions that are not self-transitory.

$$\forall \{\boldsymbol{\xi}_{t-1}, \boldsymbol{\xi}_t\} \;\Rightarrow c_{i,j} = c_{i,j} + 1, \qquad i \neq j$$
$$a_{i,j} = \frac{c_{i,j}}{\sum_{j=1}^{K} c_{i,j}}.$$

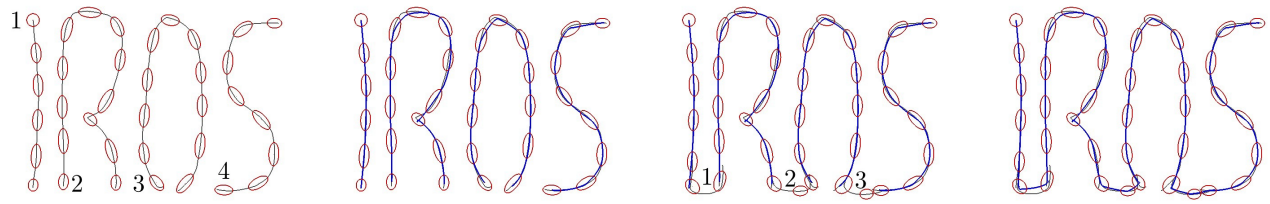
The initial emission probabilities Π_i are estimated in a similar manner, by keeping track of the starting component of each demonstration sequence.

B. Extension to HSMM

When computing the transition probabilities earlier, we only kept track of the non self-transitory instances. This is because in HMM, the self-transition probabilities $a_{i,i}$ only allow a crude implicit model of the number of iterations that we can expect to stay in a given state q_i before moving to another state. Indeed, the probability of staying d consecutive time steps in a state i follows the geometric distribution (see for example [11]) $\mathcal{P}(d) = a_{i,i}^{d-1}(1 - a_{i,i})$, decreasing exponentially with time. This is restrictive for motion generation as both spatial and temporal information is important for our task [13].

Variable duration models such as the HSMM do not use the self-transition probabilities $a_{i,i}$ of the HMM, and replaces it with an explicit model (non-parametric or parametric) of the relative time during which one stays in each state [16], [12]. For example, a univariate Gaussian distribution $\mathcal{N}(\mu_i^{\mathcal{D}}, \Sigma_i^{\mathcal{D}})$ can be used to model this duration.

In our approach, similarly to the transition probabilities calculation, we can bypass the computationally expensive batch training procedure and replace it with an online approach by keeping statistics over the state transitions. This way, as demonstrations are being performed and the underlying GMM is being built, we keep track of each state duration and accordingly update the statistics of each state. This is done using a running statistics method to compute the mean and variance for each state duration. This requires that we only keep track of the total number of samples while we incrementally add new values. Consequently, our approach does not need to store any datapoint while all learning is performed online. The overall online learning algorithm is detailed in Alg. 1.



(a) Demonstrations (b) Sampled motions (c) Incremental additions (d) Single motion

Fig. 3. A planar drawing example that highlights the flexibility of our approach. *a*) Four motions are provided as demonstrations, spelling '*IROS*'. The position of the numbers indicates the start of each demonstration. *b*) The learned model is sampled and generated motions are shown in blue. *c*) Three new demonstrations are incrementally added to the model, effectively joining the previous demonstrations to construct a single motion. *d*) A single motion can now be sampled, that was build by incrementally incorporating the set of demonstrations. The whole movement is learned by piecewise demonstration without having to rely on time re-alignment preprocessing such as dynamic time warping (DTW).

C. Sampling

Sampling from the model can take two forms: 1) deterministic, where the most likely sequence of states is sampled and remains unchanged in successive sampling trials, and 2) stochastic, where the sequence of states is sampled in a probabilistic manner. By stochastic sampling, motions that contain different options, i.e., motions that do not evolve only on a single path – for example [17], can also be represented.

In this resulting HSMM, the probability of a datapoint $\boldsymbol{\xi}_t$ to be in state i at time step t given the partial observation $\{\boldsymbol{\xi}_1, \boldsymbol{\xi}_2, \dots, \boldsymbol{\xi}_t\}$ can be recursively computed with (see for example [11])

$$\alpha_{i,t}^{\text{HSMM}} = \sum_{j=1}^{K} \sum_{d=1}^{d^{\max}} \alpha_{j,t-d}^{\text{HSMM}}\, a_{j,i}\, \mathcal{N}_{d,i}^{\mathcal{D}} \prod_{s=t-d+1}^{t} \mathcal{N}_{s,i}, \quad \text{where}$$
$$\mathcal{N}_{d,i}^{\mathcal{D}} = \mathcal{N}(d|\mu_i^{\mathcal{D}}, \Sigma_i^{\mathcal{D}}) \text{ and } \mathcal{N}_{s,i} = \mathcal{N}\big(\boldsymbol{\xi}_s|\ \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i\big).$$

For $t < d^{\max}$, the initialization is given by

$$\alpha_{i,1}^{\text{HSMM}} = \Pi_i \mathcal{N}_{1,i}^{\mathcal{D}}\, \mathcal{N}_{1,i},$$
$$\alpha_{i,2}^{\text{HSMM}} = \Pi_i \mathcal{N}_{2,i}^{\mathcal{D}} \prod_{s=1}^{2} \mathcal{N}_{s,i} + \sum_{j=1}^{K} \alpha_{j,1}^{\text{HSMM}}\, a_{j,i} \mathcal{N}_{1,i}^{\mathcal{D}} \mathcal{N}_{2,i},$$
$$\alpha_{i,3}^{\text{HSMM}} = \Pi_i \mathcal{N}_{3,i}^{\mathcal{D}} \prod_{s=1}^{3} \mathcal{N}_{s,i} + \sum_{j=1}^{K} \sum_{d=1}^{2} \alpha_{j,3-d}^{\text{HSMM}}\, a_{j,i} \mathcal{N}_{d,i}^{\mathcal{D}} \prod_{s=4-d}^{3} \mathcal{N}_{s,i},$$

etc., which corresponds to the update rule

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

Wiht this representation, a generative process can be constructed by setting equal observation probability $\mathcal{N}_{s,i} = 1 \forall i$. Note that the above iterations can be reformulated for efficient computation, see [12], [18] for details.

D. Motion Generation

By stochastically sampling from the HSMM for T time steps, we obtain a sequence of states to be visited $q_1 \dots q_T$. The step-wise reference trajectory $\mathcal{N}(\hat{\boldsymbol{\mu}}_{q_t}, \hat{\boldsymbol{\Sigma}}_{q_t})$ can be smoothly tracked by using an infinite-horizon linear quadratic regulator with a double integrator system [19]. The cost function to minimize at each time step t_0 is given by

$$c(\boldsymbol{\xi}_t, \boldsymbol{u}_t) = \sum_{t=t_0}^{\infty} (\boldsymbol{\xi}_t - \hat{\boldsymbol{\mu}}_{q_{t_0}})^\top \boldsymbol{Q}_{t_0} (\boldsymbol{\xi}_t - \hat{\boldsymbol{\mu}}_{q_{t_0}}) + \boldsymbol{u}_t^\top \boldsymbol{R} \boldsymbol{u}_t, \tag{2}$$

where $\boldsymbol{u}_t \in \mathbb{R}^m$ is the control input of the system. Setting $\boldsymbol{Q}_t = \hat{\boldsymbol{\Sigma}}_{q_t}^{-1}, \boldsymbol{R} \succ 0, \boldsymbol{\xi}_t = [\boldsymbol{x}_t{}^\top\ \dot{\boldsymbol{x}}_t^\top]^\top$, $\hat{\boldsymbol{\mu}}_{q_t} = [\hat{\boldsymbol{\mu}}_t^{x^\top}\ \hat{\boldsymbol{\mu}}_t^{\dot{x}^\top}]^\top$ with $\boldsymbol{x}, \dot{\boldsymbol{x}}$ representing the position and velocity of the system, the optimal control input $\boldsymbol{u}_t^*$ obtained by solving the algebraic Riccati equation is given by

$$\boldsymbol{u}_t^* = \boldsymbol{K}_t^{\mathcal{P}}(\hat{\boldsymbol{\mu}}_t^x - \boldsymbol{x}_t) + \boldsymbol{K}_t^{\mathcal{V}}(\hat{\boldsymbol{\mu}}_t^{\dot{x}} - \dot{\boldsymbol{x}}_t), \tag{3}$$

where $\boldsymbol{K}_t^{\mathcal{P}}$ and $\boldsymbol{K}_t^{\mathcal{V}}$ are the full stiffness and damping matrices that are regulated in following the reference trajectory of the underlying task.

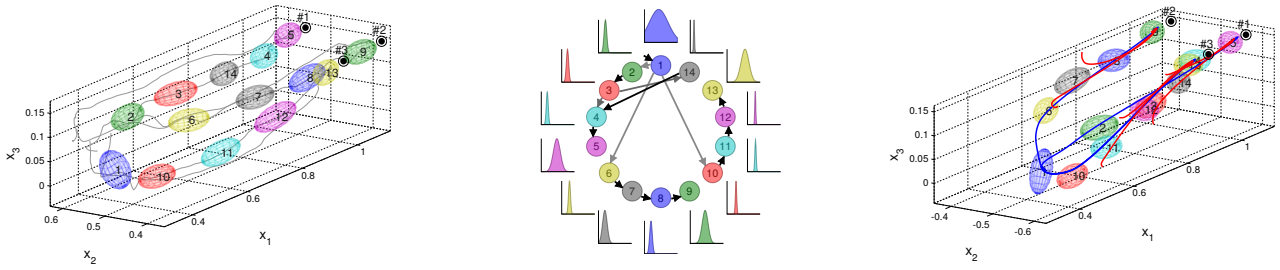
IV. Planar Drawing Example

We begin with a toy example to illustrate the advantages of our approach. We use a 2-dimensional drawing/writing skill as shown in Fig. 3.

We incrementally demonstrate a sequence of letters, as shown in Fig. 3(a). These demonstrations are in turn represented by an increasing number of Gaussians that are added to model the new datapoints. Note that the Gaussian mixture model parameters, the transition probabilities, and the duration probability distributions are all estimated online, thereby, avoiding any storage of the incoming datapoints.

Next the model is sampled stochastically and motions are generated for each of the encoded letters. These are shown in Fig. 3(b) in solid blue lines. After that, 3 new demonstrations are incrementally presented to the model (see Fig. 3(c)). The new demonstrations serve the purpose of connecting the previous demonstrations. New Gaussians are added as a result and the model parameters are updated accordingly. After this, sampling of the model can produce a single motion that contains all previous demonstrations (Fig. 3(d)), while it was incrementally built online and interactively.

This highlights the flexibility of our approach as new demonstrations can be added at will and online, without the need to store any data or batch-retrain the model. This can be particularly advantageous in a teleoperation setting as for example adding new skills on-the-fly s is a desirable feature.



(a) Demonstrations & HSMM model (b) Transition & duration probabilities (c) Generated motions

Fig. 4. Overview of experimental trials of hot-stabbing with the Baxter robot. *a*) The six demonstrated motions in grey and the learned GMM. The ellipsoids depict the equiprobable contour of one standard deviation. Note that this corresponds to the left arm of the robot, the operator's side. *b*) The transition and durations probabilities for each state. Note that only non-zero transitions are drawn while darker arrows represent larger probabilities. The duration probabilities of each state are shown accordingly. *c*) Generated motions on the teleoperation side, the right arm of Baxter. Note that the GMM is mirrored. The generated motions appear as solid blue lines. #1, #2 and #3 depict the locations of the hot-stab receptacles. Red solid lines represent motions that start from randomized initial states. These simulate the effect of communication breakdown as an operator is teleoperating the arm. Once the communication is down, the learned model can continue with the execution of the task and successfully perform the hot-stabbing.

V. Experimental Setup

We use the two-armed Baxter robot as a working example of a teleoperation system. Each of Baxter's arms has 7 DoFs, actuated by series-elastic actuators, enabling force/torque control of the joints and the end-effector. We control Baxter's arms by compensating the effect of gravity, allowing us to demonstrate motions kinesthetically. We use the left arm as the operator's side and the right arm as the teleoperated end. The left arm is used for kinesthetic demonstrations of the skill, while the right arm reproduces the motions that the learned model generates [2]. This is used as a mock-up platform within the DexROV project, where the arm will later in the project be replaced by the ROV manipulator, and the other arm will be replaced by an exoskeleton worn by the teleoperator.

To demonstrate our approach, we chose one of the most frequently executed tasks in underwater ROVs [20]. This is the task of inserting a hot-stab plug to a hot-stab receptacle as shown in Fig. 5. Hot-stabbing is used to provide hydraulic actuation to most of the tools used in underwater facilities. Hot-stabbing in the general case shares similarities with the standard peg-in-the-hole task in robotics applications, that we further simplify to a reaching task for these trials (see Fig. 1 for our hot-stabbing mock-up).

VI. Experimental Results

A user guides/teleoperates the robot, kinesthetically demonstrating the hot-stabbing skill. 6 hot-stabbing demonstrations – 2 for each goal – are provided, starting from a similar neutral joint angle configuration and reaching three previously defined goal positions on the hot-stab panel mock-up as shown in Fig. 1.

Fig. 5. CAD model of a single-port high-flow standardised hot-stab and and hot-stab receptacle. (Adapted from [21].)

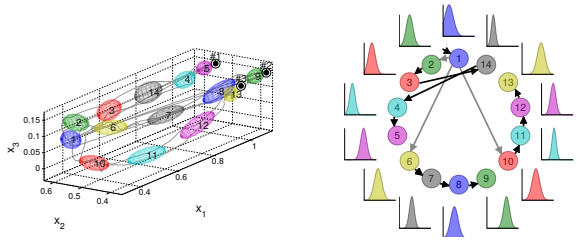


Fig. 7. Batch HSMM learning with EM yields a similar model to the one learned online.

As the demonstrations are performed, the model of the skill is being built. Fig. 4(a) shows the outcome of the online learning procedure with the demonstrations shown in grey. As new demonstrations are made available, the model incrementally grows. The transition probabilities $a_{i,j}$ and the state duration parameters $\mu_i^{\mathcal{D}}$ and $\Sigma_i^{\mathcal{D}}$ are incrementally updated accordingly.

In order to evaluate the performance of the online learning estimate, we collected the data used during the online learning. We then trained a model using batch learning with EM, based on the collected data. The results are presented in Fig. 7. We can see that the online learning estimate is very close to the local optimum found by EM.

In the hot-stabbing experiment, a model with 14 Gaussians was learned. Fig. 4(a) presents the learned model is shown along with the six demonstrated motions in grey. We can see that the Gaussians follow closely the evolution of the demonstrated motions. Motions start around the area occupied by the Gaussian numbered 1 and transition to either state 2, 10 or 6 as shown in Fig. 4(b). After this, motions evolve mainly through 3 distinct paths. The demonstrated motions reach

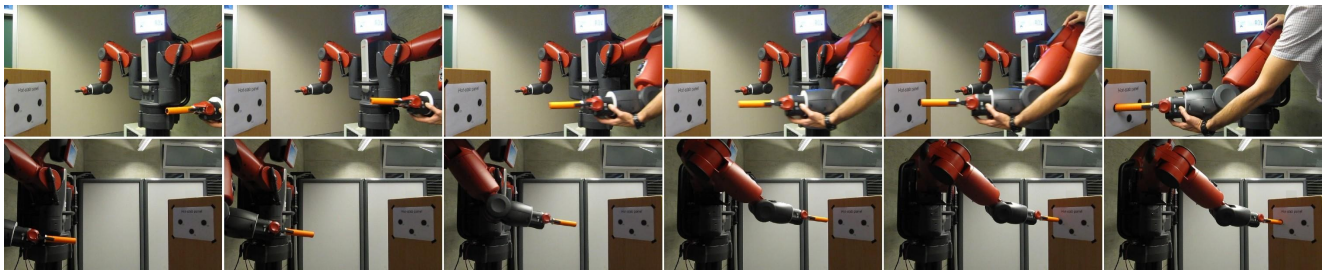


Fig. 6. Kinesthetic teaching and motion generation for the hot-stabbing skill. *Top row:* The left arm of Baxter is used to teach the hot-stabbing motion. *Bottom row:* Execution of the hot-stabbing motion, using the learned HSMM model, on the teleoperated side (i.e. Baxter's right arm).

their end goals at states 5, 9 and 13, accordingly reaching the receptacle goals numbered 1, 2 and 3. Note that the duration probabilities of the start and end states are in general larger in comparison to states that are traversed along the motions. This is accurately represented in Fig. 4(b), showing the graph connectivity and the duration probabilities for each state.

The model parameters are then exploited on the reproduction side for motion generation. The subsequent motion control is handled locally on the remote side, while only the model parameters need to be communicated at intermittent time. Once this is done, the remote arm (right) starts to follow the generated motion. The computed varying stiffness and damping profiles of the controller allow the task to be regulated in accordance with the required precision. In essence this allows us to control *lazily* along task directions that do not matter and *accurately* along the important task directions, by following a minimal intervention principle [22], see [2] for details.

Three motions starting from a neutral position and generated for the three mock-up receptacles are shown in Fig. 4(c) in blue. In red, we show motions that start from random initial positions along a hot-stabbing movement. This is used to simulate a failure in communication as the operator directly teleoperates the remote arm. As soon as the communication is lost, our system samples the learned model, starting from the current state, and generates a motion that continues the execution of the hot-stabbing task. Note that the model here is mirrored for the teleoperated side, on the right arm of the robot. Snapshots of the accompanying video[1] are presented in Fig. 6.

A. Evaluation

To evaluate the efficiency of the learned model we compare the end position of the generated hot-stabbing motions against the demonstrations. We chose this metric as the position of the hot-stabbing plug at the end of the motion should reach the receptacle entrance –up to some allowed variance– while the path to this state can vary reasonably. This is also apparent in the demonstrated motions (Fig. 4(a)).

We calculate the average end state from the set of demonstrations per receptacle. We assume this as the ground truth against which the generated motions' outcomes are compared. For each target receptacle, we sample 10 motions from the learned model starting from random initial states. We report also the variance of the demonstrations per receptacle for comparison. Fig. 8 shows the 30 hot-stabbing motions, 10 per receptacle target, that were generated by sampling the learned model, beginning at randomized initial states.

Table I summarises the results of the evaluation trials. From the RMSEs of the demonstrations we see that there is only small variance in the end point of the kinesthetic teaching motions. The motions that are generated by the learned model are similarly accurate. We see that the hot-stabbing motions accurately position the plug according to the demonstrations. What is more, the outcome is highly repeatable, as the very low variance value demonstrates. In practice all motions to a particular receptacle converge to the same end-point at the end of the motion regardless of the starting state. As all reproduction RMSEs are bellow 1cm, we conclude that all 30 trials are successful in hot-plugging to the different receptacle targets.

VII. Conclusion

In this paper, we presented an online and incrementally learned HSMM representation for encoding manipulation tasks in semi-autonomous underwater teleoperation scenarios. We demonstrated how such representation can be learned from incremental demonstrations, without the need to store demonstration data, and how motions can be regenerated from the learned model. We evaluated the performance of

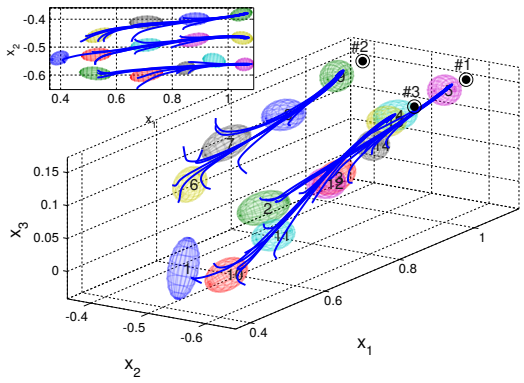


Fig. 8. Evaluation trials with 30 motions (10 per hot-stabbing receptacle, drawn in blue lines), starting from randomized initial states.

[1]Available at https://youtu.be/QsZkiTKh5DY.

TABLE I

RMSEs OF MULTIPLE AVERAGED TRIALS, SEE SECTION VI-A FOR DETAILS AND DISCUSSION.

Receptacle Number	#1	#2	#3
Reproductions, RMSE	0.77cm	0.74cm	0.99cm

our approach with a common ROV task and showed how a learned model can reproduce motions with high accuracy and repeatability. With the proposed approach, only a small set of model parameters needs to be communicated from the operator side to the teleoperated system, rendering our approach robust to intermittent communication and latency.

In future work we aim to extend our model to multiple skill primitives that can be assembled in series and in parallel. Further work will investigate how the trained model can be used to classify the operator's intention and communicate the appropriate skill activation to the teleoperated side. This way we will be able to use the same learned model for skill recognition and classification on one side, and on the other side use it for semi-autonomous control of the robot, exploiting the synthesis capability of the generative model.

ACKNOWLEDGEMENTS

The authors would like to thank Daniel Berio for his help with the experimental trials. Source codes of all presented work will be provided at the time of publication.

REFERENCES

[1] J. Gancet, D. Urbina, P. Letier, M. Ilzokvitz, P. Weiss, F. Gauch, G. Antonelli, G. Indiveri, G. Casalino, A. Birk, M. F. Pfingsthorn, S. Calinon, A. Tanwani, A. Turetta, C. Walen, and L. Guilpain, "DexROV: Dexterous undersea inspection and maintenance in presence of communication latencies," in *IFAC Workshop on Navigation, Guidance and Control of Underwater Vehicles (NGCUV)*, Girona, Spain, 2015,

[2] A. Tanwani and S. Calinon, "Learning robot manipulation tasks with task-parameterized semitied hidden semi-markov model," *Robotics and Automation Letters, IEEE*, vol. 1, no. 1, pp. 235–242, Jan 2016,

[3] B. Kulis and M. I. Jordan, "Revisiting k-means: New algorithms via Bayesian nonparametrics," in *Proc. Intl Conf. on Machine Learning (ICML)*, Edinburgh, Scotland, UK, 2012, pp. 1–8,

[4] A. Ijspeert, J. Nakanishi, P. Pastor, H. Hoffmann, and S. Schaal, "Dynamical movement primitives: Learning attractor models for motor behaviors," *Neural Computation*, vol. 25, no. 2, pp. 328–373, 2013,

[5] S. Calinon, Z. Li, T. Alizadeh, N. G. Tsagarakis, and D. G. Caldwell, "Statistical dynamical systems for skills acquisition in humanoids," in *Proc. IEEE Intl Conf. on Humanoid Robots (Humanoids)*, Osaka, Japan, 2012, pp. 323–329,

[6] D. Lee and C. Ott, "Incremental motion primitive learning by physical coaching using impedance control," in *Proc. IEEE/RSJ Intl Conf. on Intelligent Robots and Systems (IROS)*, Taipei, Taiwan, October 2010, pp. 4133–4140,

[7] D. Lee, C. Ott, and Y. Nakamura, "Mimetic communication model with compliant physical contact in human-humanoid interaction," *Intl Journal of Robotics Research*, vol. 29, no. 13, pp. 1684–1704, 2010,

[8] A. Chan, E. A. Croft, and J. J. Little, "Modeling nonconvex workspace constraints from diverse demonstration sets for constrained manipulator visual servoing," in *Proc. IEEE Intl Conf. on Robotics and Automation (ICRA)*, May 2013, pp. 3062–3068,

[9] C. Bowen and R. Alterovitz, "Closed-loop global motion planning for reactive execution of learned tasks," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2014,

[10] D. Kulic, W. Takano, and Y. Nakamura, "Incremental learning, clustering and hierarchy formation of whole body motion patterns using adaptive hidden markov chains," *Intl Journal of Robotics Research*, vol. 27, no. 7, pp. 761–784, 2008,

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

[12] S.-Z. Yu and H. Kobayashi, "Practical implementation of an efficient forward-backward algorithm for an explicit-duration hidden Markov model," *IEEE Trans. on Signal Processing*, vol. 54, no. 5, pp. 1947–1951, 2006,

[13] 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 Intl Conf. on Intelligent Robots and Systems (IROS)*, San Francisco, CA, USA, September 2011, pp. 3413–3418,

[14] A. Roychowdhury, K. Jiang, and B. Kulis, "Small-variance asymptotics for hidden markov models," in *Advances in Neural Information Processing Systems (NIPS)*, 2013, pp. 2103–2111,

[15] J. Gauvain and C.-H. Lee, "Maximum a posteriori estimation for multivariate gaussian mixture observations of markov chains," *Speech and Audio Processing, IEEE Transactions on*, vol. 2, no. 2, pp. 291–298, Apr 1994,

[16] H. Zen, K. Tokuda, T. Masuko, T. Kobayasih, and T. Kitamura, "A hidden semi-Markov model-based speech synthesis system," *IEICE Trans. on Information and Systems*, vol. E90-D, no. 5, pp. 825–834, May 2007,

[17] G. Ganesh and E. Burdet, "Motor planning explains human behaviour in tasks with multiple solutions," *Robotics and Autonomous Systems*, vol. 61, no. 4, pp. 362–368, 2013,

[18] S.-Z. Yu, "Hidden semi-markov models," *Artificial Intelligence*, vol. 174, pp. 215–243, 2010,

[19] S. Calinon, D. Bruno, and D. G. Caldwell, "A task-parameterized probabilistic model with minimal intervention control," in *Proc. IEEE Intl Conf. on Robotics and Automation (ICRA)*, Hong Kong, China, May-June 2014, pp. 3339–3344,

[20] A. Bleicher, "Gulf spill one year later: Lessons for robotics," Website: http://spectrum.ieee.org/robotics/industrial-robots/gulf-spill-one-year-later-lessons-for-robotics, 2015,

[21] T. M. Technology, "http://www.tmtrov.com.au/tooling/standard-tooling/tmt-single-port-high-flow-hot-stab," Website, 2015,

[22] E. Todorov and M. I. Jordan, "A minimal intervention principle for coordinated movement," in *Advances in Neural Information Processing Systems (NIPS)*, 2002, pp. 27–34,