Article

The International Journal of Robotics Research
2021, Vol. 40(10-11) 1306–1325
© The Author(s) 2021
Article reuse guidelines:
sagepub.com/journals-permissions
DOI: 10.1177/02783649211032721
journals.sagepub.com/home/ijr

Sequential robot imitation learning from observations

Ajay Kumar Tanwani[1], Andy Yan[1], Jonathan Lee[1], Sylvain Calinon[2] and Ken Goldberg[1]

Abstract

This paper presents a framework to learn the sequential structure in the demonstrations for robot imitation learning. We first present a family of task-parameterized hidden semi-Markov models that extracts invariant segments (also called subgoals or options) from demonstrated trajectories, and optimally follows the sampled sequence of states from the model with a linear quadratic tracking controller. We then extend the concept to learning invariant segments from visual observations that are sequenced together for robot imitation. We present Motion2Vec that learns a deep embedding space by minimizing a metric learning loss in a Siamese network: images from the same action segment are pulled together while being pushed away from randomly sampled images of other segments, and a time contrastive loss is used to preserve the temporal ordering of the images. The trained embeddings are segmented with a recurrent neural network, and subsequently used for decoding the end-effector pose of the robot. We first show its application to a pick-and-place task with the Baxter robot while avoiding a moving obstacle from four kinesthetic demonstrations only, followed by suturing task imitation from publicly available suturing videos of the JIGSAWS dataset with state-of-the-art 85.5% *segmentation accuracy and* 0.94 *cm error in position per observation on the test set.*

Keywords

Hidden semi-Markov model, robot learning, imitation learning, learning and adaptive systems

1. Introduction

Imitation learning provides a promising approach to teach new robotic manipulation skills from expert demonstrations (Argall et al., 2009; Billard et al., 2016; Schaal et al., 2003). Generalizing robot manipulation skills to new situations requires extracting disentangled representations from demonstrations that capture the relationships between objects and the environment while being invariant to lighting, background, and other geometric properties such as position, size of external objects, and viewpoint of the camera. When the demonstrations have sequential, recursive, relational, or other kinds of structure, structured representations can be useful to infer meaningful hidden associations for learning of manipulation skills.

Consider, for example, a frequently used *pick-and-place* task in robotics applications that may be decomposed into temporally connected shorter segments such as *reach*, *grasp*, *move*, *drop*, and so on. Similarly, the surgical suturing task may be decomposed into temporally connected movement primitives or action segments such as *needle insertion*, *needle extraction*, *needle hand-off*, etc. Such a hierarchical decomposition is analogous to the speech recognition and synthesis problem where words and sentences are synthesized from phoneme- and triphone-based segments (Rabiner, 1989; Zhang et al., 2017). Recent trends in imitation leaning are forgoing such a task structure for end-to-end supervised learning which requires a large-scale collection and labeling of training demonstrations.

The focus of this paper is on exploiting the sequential structure in the demonstrations by extracting invariant segments and sequencing them together for robot imitation. We first present a family of task-parameterized hidden semi-Markov models (HSMMs) (Rabiner, 1989; Tanwani, 2018; Yu, 2010) for spatiotemporal encoding of the demonstrations, and combine it with a linear quadratic tracking (LQT) controller (Borrelli et al., 2011) to

[1]University of California, Berkeley, CA, USA
[2]Idiap Research Institute, Valais, Switzerland

Corresponding author:
Ajay Kumar Tanwani, University of California, Berkeley, 2111 Etcheverry Hall, 2505 Hearst Avenue, Berkeley, CA 94709, USA.
Email: ajay.tanwani@berkeley.edu

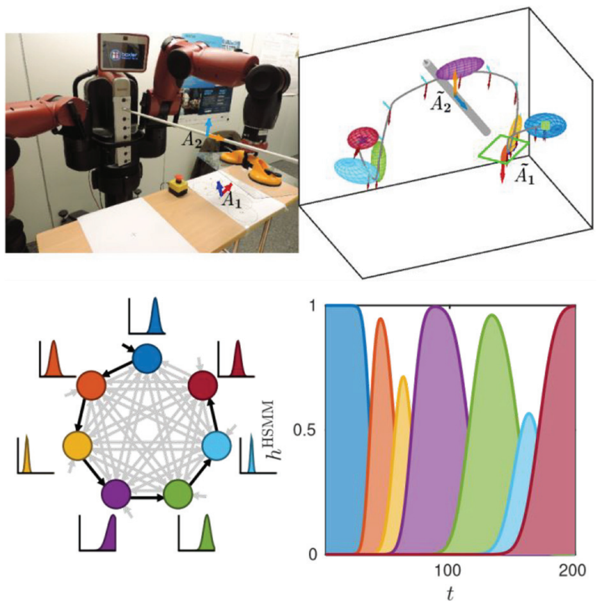


Fig. 1. Top-left: Baxter robot picks the glass plate with a suction lever and places it on the cross after avoiding an obstacle of varying height. Top-right: reproduction for previously unseen object and obstacle position. Here $\{\tilde{A}_1, \tilde{b}_1\}$ and $\{\tilde{A}_2, \tilde{b}_2\}$ indicate the reference frame positions attached to the glass plate (green rectangle) and the obstacle (gray rod), respectively. Ellipsoids of different colors represent the 3D cross-sections of the Gaussians that encode the observation distributions. Bottom-left: left–right HSMM encoding of the task with duration model shown next to each state. Bottom-right: evolution of the forward variable of HSMM over time.

reproduce the robot movement. The task-parameterized formulation systematically adapts the generated trajectories to changing situations such as pose/size of the objects in the environment (Calinon, 2016; Tanwani and Calinon, 2016; Wilson and Bobick, 1999).

We then present Motion2Vec for acquiring motion-centric representations of manipulation skills from videos for imitation learning (Tanwani et al., 2020). We use metric learning with a Siamese network to bring similar action segments, images with same discrete labels, together in an embedding space, while preserving the temporal ordering in the embedded observations. Motion2Vec moves the video observations into a low-dimensional vector domain where closeness refers to spatiotemporal grouping of the same action segments.

We show two applications of sequential robot imitation learning: (1) pick-and-place an object while avoiding a moving obstacle with the Baxter robot using four kinesthetic demonstrations only (see Figure 1); (2) segmentation and imitation of surgical suturing motions on the dual-arm da Vinci robot from publicly available videos of the JIGSAWS dataset (see Figure 2), while obtaining better segmentation accuracy of 85.5% on the leave one super trial out test set than reported in the literature, e.g., DiPietro et al. (2016) and Lea et al. (2016) report accuracy of 83.3% and 81.4%, respectively.

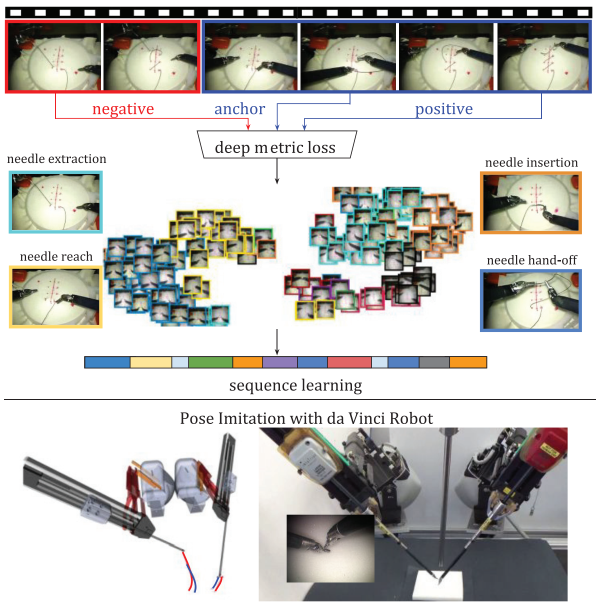


Fig. 2. Motion2Vec groups similar action segments together in an embedding space. The embedding space is represented above with a t-distributed stochastic neighbor embedding (t-SNE) plot. Action segments are obtained by sequence learning with a recurrent neural network (RNN) for a given parametrization of the Siamese network. The learned representation is applied to surgical suturing segmentation and pose imitation in simulation and real on the da Vinci robot.

This paper unifies and extends our previous work on encoding manipulation skills in a task-adaptive manner from trajectories and videos (Tanwani et al., 2018, 2020). The main contributions are as follows.

1. A task-parameterized generative model that combines the HSMM with a LQT controller for imitation learning from kinesthetic demonstrations.
2. A representation learning approach for spatiotemporal alignment of action segments/sub-goals/options in an embedding space from video observations that is invariant to nuisance variables such as lightning, background, and camera viewpoint.
3. Learning manipulation skills for a pick-and-place task while avoiding a moving obstacle from a few kinesthetic demonstrations, and for a vision-based suturing task with a da Vinci robot from the publicly available JIGSAWS dataset with performance improvement over state-of-the-art methods.

1.1. Organization of the paper

We give a brief overview of imitation learning approaches in Section 2. Section 3 describes our sequential learning from trajectories approach followed by task-parameterized formulations in Section 4. In Section 5, we present the imitation learning from videos approach and then evaluate the performance on pick-and-place with the Baxter robot and surgical suturing with the da Vinci robot in Section 6. Finally, we conclude the paper with an outlook to our future work.

2. Related work

The two main challenges in imitation learning include (Nehaniv and Dautenhahn, 2004): (1) *what to learn*, acquiring meaningful features of the task from demonstrations for imitation; and (2) *how to learn*, learning a control policy from the features to reproduce the demonstrated behavior. Common approaches to imitation learning include *behavior cloning* and *inverse reinforcement learning* (IRL) with dynamic movement primitives (Ijspeert et al., 2013), Gaussian mixture models (Calinon and Lee, 2018), task-parametrized generative models (Tanwani, 2018), generative adversarial imitation learning (Ho and Ermon, 2016), one-shot imitation learning (Duan et al., 2017), Dagger (Ross et al., 2011), and behavior cloning from observation (Torabi et al., 2018) (see Osa et al. (2018) for an overview). In contrast to direct trajectory learning from demonstrations, we focus on the contextual understanding by exploiting the sequential structure in the demonstrations.

2.1. Sequential learning from trajectories

Hidden Markov models (HMMs) have been typically used as a generative model for recognition and generation of movement skills in robotics (Asfour et al., 2008; Duque et al., 2019; Lee and Ott, 2010; Mor et al., 2021; Rozo et al., 2020; Tanwani and Calinon, 2017; Vakanski et al., 2012). For example, Kulic et al. (2008) used HMMs to incrementally group whole-body motions based on their relative distance in HMM space. Lee and Ott (2010) presented an iterative motion primitive refinement approach with HMMs. Niekum et al. (2012) used the beta process autoregressive HMM for learning from unstructured demonstrations. Figueroa and Billard (2017) used the transformation invariant covariance matrix for encoding tasks with a Bayesian non-parametric HMM. Yang et al. (2019) and Pignat and Calinon (2017) combined HSMM with Gaussian mixture regression for assisted teleoperation and dressing. Conditional random fields (CRFs), on the other hand, directly encode the conditional probability distribution of the output labels given the input observation sequence in a discriminative manner (Lafferty et al., 2001; Sutton and McCallum, 2012). The interested reader can find connections of probabilistic inference methods for planning and control in Todorov (2008), Toussaint (2009), Kappen et al. (2012), and Levine (2018). Other related application contexts in imitation learning include options framework (Fox et al., 2017; Krishnan et al., 2017), sequencing primitives (Medina R. and Billard, 2017), temporal alignment (Bowen and Alterovitz, 2020; Shiarlis et al., 2018), intention recognition (Tanwani and Calinon, 2017; Ti et al., 2019), and neural task programs (Fox et al., 2018; Xu et al., 2017). This paper emphasizes learning manipulation skills from human demonstrations using a family of HMMs by sequencing the atomic movement segments or primitives. We build upon these works to present a task-parameterized HSMM for segmentation, recognition, and synthesis of robot movement. We observe the demonstrations with respect to different coordinate systems describing virtual landmarks or objects of interest, and adapt the model according to the environmental changes in a systematic manner. Capturing such invariant representations allows us to compactly encode the task variations than using a standard regression problem.

2.2. Weakly supervised learning from videos

Acquiring robot manipulation skills from videos by imitation provides a scalable alternative to traditional kinesthetic and teleoperation interfaces. However, uncontrolled variables such as lighting, background and camera viewpoint pose a challenge to robot learning from video observations. Learning from multiple viewpoints, temporal sequences, labeled action segments, weakly supervised signals such as order of sub-actions, text-based annotations or unsupervised learning are feasible alternatives to learning dense pixel-wise visual descriptors from observations (Florence et al., 2018; Schmidt et al., 2017; Zeng et al., 2017). Kuehne and Serre (2015) presented a generative framework for end-to-end action recognition by extracting Fisher vectors from videos and sequencing them with HMMs, followed by a weakly supervised approach for temporal action segmentation with RNN–HMM (Kuehne et al., 2019). Tang (2012) learn temporal structure in the videos for complex event detection. Liu et al. (2017) learn a translation invariant policy between the expert and the learner contexts. Doersch et al. (2015) use spatial coherence in the neighboring pixels for learning unsupervised visual representations. Misra et al. (2016) proposed shuffle and learn to maintain temporal order in learning the visual representation. Wang and Gupta (2015) used triplet loss to encourage the first and the last frame of the same video together in the embedding space, while pushing away the negative sample from another class. Sermanet et al. (2017) used metric learning loss for getting a temporally coherent viewpoint invariant embedding space with multi-view or single-view images and used nearest neighbors in the embedding space for imitation learning. Dwibedi et al. (2018) extended the approach to multiple frames, and use a temporal cycle consistency (TCC) loss by matching frames over time across videos in Dwibedi et al. (2019). Finn and Levine (2016) and Ebert et al. (2018) presented a deep action-conditioned visual foresight model with model-predictive control for learning from pixels directly. Visual imitation is gaining traction in several emerging robotics applications (Finn et al., 2017; Hoque et al., 2020; Sundaresan et al., 2020; Young et al., 2020).

Siamese networks learn a similarity function across images in an embedding space (Hermans et al., 2017; Koch et al., 2015; Schroff et al., 2015). In this paper, we encode the video observations based on spatiotemporal alignment of action segments/options, in addition to using time only with self-supervised representations (Sermanet et al., 2017). We use metric learning with triplet loss to attract similar actions in an embedding space and segment the resulting embedding space with a hybrid deep neural network (DNN)–RNN model. We apply the approach for downstream tasks of learning action segments/surgemes and end-effector poses on the dual-arm da Vinci robot from publicly available suturing videos of the JIGSAWS dataset (Gao et al., 2014). Results suggest performance improvement in segmentation over state-of-the-art baselines (Ahmidi et al., 2017; DiPietro et al., 2016), while introducing pose imitation on this dataset with 0.94 cm error in position per observation, respectively.

3. Imitation from trajectories

Let $\{\boldsymbol{\xi}_t\}_{t=1}^T$ denote the sequence of observations with $\boldsymbol{\xi}_t \in \mathbb{R}^D$ collected while demonstrating a task. The observations may represent the kinesthetic data such as the pose and the velocities of the end-effector of the human arm, haptic information, or any arbitrary features defining the task variables of the environment. The observation sequence is associated with an unknown hidden state sequence $\{z_t\}_{t=1}^T$ with $z_t \in \{1 \dots K\}$ belonging to the discrete set of K hidden states. For example, the hidden states may correspond to different segments of the task such as reach, grasp, move, etc. The transition between one segment i to another segment j is denoted by the transition matrix $\boldsymbol{a} \in \mathbb{R}^{K \times K}$ with $a_{i,j} \triangleq P(z_t = j | z_{t-1} = i)$. The parameters $\{\mu_j^S, \Sigma_j^S\}$ represent the mean and the standard deviation of staying s consecutive time steps in state j as $p(s)$ estimated by a Gaussian $\mathcal{N}(s|\mu_j^S, \Sigma_j^S)$. The hidden state follows a categorical distribution with $z_t \sim \text{Cat}(\boldsymbol{\pi}_{z_{t-1}})$ where $\boldsymbol{\pi}_{z_{t-1}} \in \mathbb{R}^K$ is the next state transition distribution over state z_{t-1} with Π_i as the initial probability, and the observation $\boldsymbol{\xi}_t$ is drawn from the output distribution of state j, described by a multivariate Gaussian with parameters $\{\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j\}$. The overall parameter set is defined by $\boldsymbol{\Theta} = \{\Pi_i, \{a_{i,m}\}_{m=1}^K, \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i, \mu_i^S, \Sigma_i^S\}_{i=1}^K$. We seek to model and infer the semantic structure of the hidden state sequence $\{z_t\}_{t=1}^T$ from the observations. We learn the joint probability density of the observation and the hidden state sequence $\mathcal{P}(\boldsymbol{\xi}_t, z_t)$, where the data generating emission distribution $\mathcal{P}(\boldsymbol{\xi}_t | z_t)$ is simplified for exact inference of the model structure $\mathcal{P}(z_t | \boldsymbol{\xi}_t)$.

3.1. HMMs

HMMs encapsulate the spatiotemporal information by augmenting a mixture model with latent states that sequentially evolve over time in the demonstrations (Rabiner, 1989). A HMM is thus defined as a doubly stochastic process, one with a sequence of hidden states and another with a sequence of observations/emissions. Spatiotemporal encoding with HMMs can handle movements with variable durations, recurring patterns, options in the movement, or partial/unaligned demonstrations. Without loss of generality, we present our formulation with semi-Markov models for the remainder of the paper. Semi-Markov models relax the Markovian structure of state transitions by relying not only upon the current state but also on the duration/elapsed time in the current state, i.e., the underlying process is defined by a *semi-Markov* chain with a variable duration time for each state. The state duration stay is a random integer variable that assumes values in the set $\{1, 2, \dots, s^{\max}\}$. The value corresponds to the number of observations produced in a given state, before transitioning to the next state. HSMMs associate an observable output distribution with each state in a semi-Markov chain (Yu, 2010), similar to how we associate a sequence of observations with a Markov chain in a HMM.

3.2. Encoding with HSMM

For learning and inference in a HMM (Rabiner, 1989), we make use of the intermediary variables as: (1) *forward variable*, $\alpha_{t,i}^{\text{HMM}} \triangleq P(z_t = i, \boldsymbol{\xi}_1 \dots \boldsymbol{\xi}_t | \theta)$, probability of a datapoint $\boldsymbol{\xi}_t$ to be in state i at time step t given the partial observation sequence $\{\boldsymbol{\xi}_1, \dots, \boldsymbol{\xi}_t\}$; (2) *backward variable*, $\beta_{t,i}^{\text{HMM}} \triangleq P(\boldsymbol{\xi}_{t+1} \dots \boldsymbol{\xi}_T | z_t = i, \theta)$, probability of the partial observation sequence $\{\boldsymbol{\xi}_{t+1}, \dots, \boldsymbol{\xi}_T\}$ given that we are in the i th state at time step t; (3) *smoothed node marginal* $\boldsymbol{\gamma}_{t,i}^{\text{HMM}} \triangleq P(z_t = i | \boldsymbol{\xi}_1 \dots \boldsymbol{\xi}_T, \theta)$, probability of $\boldsymbol{\xi}_t$ to be in state i at time step t given the full observation sequence $\boldsymbol{\xi}$; and (4) *smoothed edge marginal* $\zeta_{t,i,j}^{\text{HMM}} \triangleq P(z_t = i, z_{t+1} = j | \boldsymbol{\xi}_1 \dots \boldsymbol{\xi}_T, \theta)$, probability of $\boldsymbol{\xi}_t$ to be in state i at time step t and in state j at time step $t+1$ given the full observation sequence $\boldsymbol{\xi}$. The intermediary variables are mathematically represented as

$$
\begin{aligned}
\alpha_{t,i}^{\text{HMM}} &= \left(\sum_{j=1}^{K} \alpha_{t-1,j}^{\text{HMM}} \, a_{j,i} \right) \mathcal{N}(\boldsymbol{\xi}_t | \, \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i) \\
\beta_{t,i}^{\text{HMM}} &= \sum_{j=1}^{K} a_{i,j} \, \mathcal{N}(\boldsymbol{\xi}_{t+1} | \, \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j) \, \beta_{t+1,j}^{\text{HMM}} \\
\gamma_{t,i}^{\text{HMM}} &= \frac{\alpha_{t,i}^{\text{HMM}} \beta_{t,i}^{\text{HMM}}}{\sum_{k=1}^{K} \alpha_{t,k}^{\text{HMM}} \beta_{t,k}^{\text{HMM}}} \\
\zeta_{t,i,j}^{\text{HMM}} &= \frac{\alpha_{t,i}^{\text{HMM}} \, a_{i,j} \, \mathcal{N}(\boldsymbol{\xi}_{t+1} | \, \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j) \, \beta_{t+1,j}^{\text{HMM}}}{\sum_{k=1}^{K} \sum_{l=1}^{K} \alpha_{t,k}^{\text{HMM}} \, a_{k,l} \, \mathcal{N}(\boldsymbol{\xi}_{t+1} | \, \boldsymbol{\mu}_l, \boldsymbol{\Sigma}_l) \, \beta_{t+1,l}^{\text{HMM}}}
\end{aligned}
\tag{1}
$$

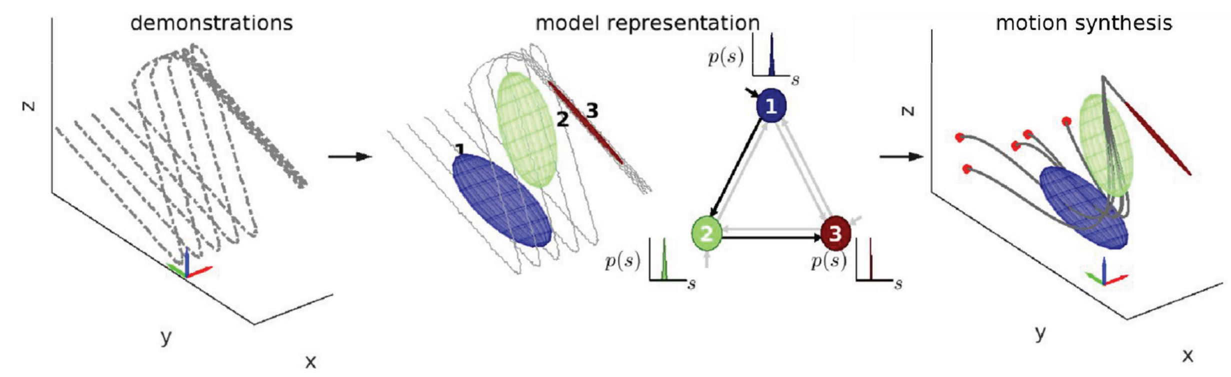


Fig. 3. Conceptual illustration of HSMM for imitation learning. Left: three-dimensional Z-shaped demonstrations composed of five equally spaced trajectory samples. Middle: demonstrations are encoded with a three-state HMM represented by Gaussians (shown as ellipsoids) that represent the blue, green, and red segments, respectively. The transition graph shows a duration model (Gaussian) next to each node. Right: the model is combined with LQT to synthesize motion in performing robot manipulation tasks from five different initial conditions marked with orange squares (see also Figure 4).

The expected complete log-likelihood of HMMs for a set of M demonstrations,

$$\mathcal{Q}(\theta, \theta^{\text{old}}) = \mathbb{E}\left\{ \sum_{m=1}^{M} \sum_{t=1}^{T} \log \mathcal{P}(\boldsymbol{\xi}_{m,t}, z_t | \theta) \mid \boldsymbol{\xi}, \theta^{\text{old}} \right\}$$

is maximized in an expectation–maximization (EM) manner with

$$\begin{aligned} \mathcal{Q}(\theta, \theta^{\text{old}}) &= \sum_{i=1}^{K} \sum_{m=1}^{M} \gamma_{m,1,i}^{\text{HMM}} \log \Pi_i \\ &+ \sum_{i=1}^{K} \sum_{j=1}^{K} \sum_{m=1}^{M} \sum_{t=1}^{T} \zeta_{m,t,i,j}^{\text{HMM}} \log a_{i,j} \\ &+ \sum_{m=1}^{M} \sum_{t=1}^{T} \sum_{i=1}^{K} \mathcal{P}(z_t{=}i | \boldsymbol{\xi}_{m,t}, \theta^{\text{old}}) \log \mathcal{N}(\boldsymbol{\xi}_{m,t} | \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i) \end{aligned} \tag{2}$$

$$E-step: \quad \gamma_{m,t,i}^{\text{HMM}} = \frac{\alpha_{t,i}^{\text{HMM}} \beta_{t,i}^{\text{HMM}}}{\sum_{k=1}^{K} \alpha_{t,k}^{\text{HMM}} \beta_{t,k}^{\text{HMM}}}$$

$$M-step: \quad \Pi_i \leftarrow \frac{\sum_{m=1}^{M} \gamma_{m,1,i}^{\text{HMM}}}{M}$$

$$a_{i,j} \leftarrow \frac{\sum_{m=1}^{M} \sum_{t=1}^{T_m-1} \zeta_{m,t,i,j}^{\text{HMM}}}{\sum_{m=1}^{M} \sum_{t=1}^{T_m-1} \gamma_{m,t,i}^{\text{HMM}}}$$

$$\boldsymbol{\mu}_i \leftarrow \frac{\sum_{m=1}^{M} \sum_{t=1}^{T_m} \gamma_{m,t,i}^{\text{HMM}} \boldsymbol{\xi}_{m,t}}{\sum_{m=1}^{M} \sum_{t=1}^{T_m} \gamma_{m,t,i}^{\text{HMM}}}$$

$$\boldsymbol{\Sigma}_i \leftarrow \frac{\sum_{m=1}^{M} \sum_{t=1}^{T_m} \gamma_{m,t,i}^{\text{HMM}} (\boldsymbol{\xi}_{m,t} - \boldsymbol{\mu}_i)(\boldsymbol{\xi}_{m,t} - \boldsymbol{\mu}_i)^{\text{T}}}{\sum_{m=1}^{M} \sum_{t=1}^{T_m} \gamma_{m,t,i}^{\text{HMM}}}$$

Note that numerical underflow issues occur with a naive implementation of the above algorithm. In practice, a simple approach to avoid this issue is to rely on scaling factors during the computation of the forward and backward variables, which get canceled out when normalizing the posterior (Rabiner, 1989). Parameters $\{\Pi_i, \{a_{i,m}\}_{m=1}^{K}, \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i\}_{i=1}^{K}$ are estimated using the EM algorithm for HMMs, and the duration parameters $\{\mu_i^S, \Sigma_i^S\}_{i=1}^{K}$ are estimated empirically from the data after training using the most likely hidden state sequence $z_t = \{z_1 \dots z_T\}$. In simple words, if the computed hidden state sequence for a given observation sequence reads as $1, 1, 1, 1, 1, 3, 3, 3, 3, 3, 3, 2, 2, 2,$ the state duration mean for the three states would be 5, 6, and 3 steps (see Figure 3 for an overview of the approach).

3.3. Viterbi decoding with HSMM

The most probable sequence of the hidden states z_t^* for a given observation sequence and the model parameters is determined by the Viterbi algorithm (Rabiner, 1989; Yu, 2010). The decoding problem is

$$z_t^* = \underset{z_1 \dots z_T}{\arg\max}\; P(\boldsymbol{\xi}_1 \dots \boldsymbol{\xi}_T, z_1 \dots z_T | \boldsymbol{\Theta}) \tag{3}$$

The Viterbi algorithm employs the max operator in the forward pass, followed by a backward pass to recover the most probable path z_t^*. Let us denote $\delta_{t,i}$ as the likelihood of the most probable state sequence ending in state i based on the first t observations, recursively estimated as

$$\delta_{t,i} = \max_s \left(\max_j \delta_{t-s,j}\, a_{j,i} \right) P(s|i) \prod_{c=t-s+1}^{t} P(\boldsymbol{\xi}_t | i) \tag{4}$$

The maximum over $\delta_{T,i}$ gives the maximum likelihood estimate of the observed sequence and the terminal state z_T. Backtracking over the auxiliary variable $\psi_{t,i} = \arg\max_j (\delta_{t-1,i}\, a_{j,i}), \forall t \in \{2 \dots T\}$, gives the desired hidden state sequence. Taken over all T and K values, the Viterbi decoding takes $\mathcal{O}(K^2T^2)$.

3.4. Synthesis with HSMM

Given the learned model parameters, the probability of the observed sequence $\{\boldsymbol{\xi}_1 \ldots \boldsymbol{\xi}_t\}$ to be in a hidden state $z_t = i$ at the end of the sequence (also known as *filtering* problem) is computed with the help of the forward variable as

$$P(z_t|\boldsymbol{\xi}_1, \ldots, \boldsymbol{\xi}_t) = h_{t,i}^{\text{HMM}} = \frac{\alpha_{t,i}^{\text{HMM}}}{\sum\limits_{k=1}^{K} \alpha_{t,k}^{\text{HMM}}} = \frac{\pi_i \mathcal{N}(\boldsymbol{\xi}_t|\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)}{\sum\limits_{k=1}^{K} \pi_k \mathcal{N}(\boldsymbol{\xi}_t|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)} \tag{5}$$

Sampling from the model for predicting the sequence of states over the next time horizon $P(z_t, z_{t+1}, \ldots, z_{T_p}|\boldsymbol{\xi}_1, \ldots, \boldsymbol{\xi}_t)$ can be done in two ways. (1) Stochastic sampling: the sequence of states is sampled in a probabilistic manner given the state duration and the state transition probabilities. By stochastic sampling, motions that contain different options and do not evolve only on a single path can also be represented. Starting from the initial state $z_t = i$, the s duration steps are sampled from $\{\mu_i^S, \Sigma_i^S\}$, after which the next transition state is sampled $z_{t+s+1} \sim \boldsymbol{\pi}_{z_{t+s}}$. The procedure is repeated for the given time horizon in a receding horizon manner. (2) Deterministic sampling: the most likely sequence of states is sampled and remains unchanged in successive sampling trials. We use the forward variable of HSMM for deterministic sampling from the model. The forward variable $\alpha_{t,i}^{\text{HSMM}} \triangleq P(z_t = i, \boldsymbol{\xi}_1 \ldots \boldsymbol{\xi}_t|\theta)$ requires marginalizing over the duration steps along with all possible state sequences. The probability of a datapoint $\boldsymbol{\xi}_t$ to be in state i at time step t given the partial observation sequence $\{\boldsymbol{\xi}_1, \ldots, \boldsymbol{\xi}_t\}$ is now specified as (Yu, 2010)

$$\alpha_{t,i}^{\text{HSMM}} = \sum_{s=1}^{\min(s^{\max}, t-1)} \sum_{j=1}^{K} \alpha_{t-s,j}^{\text{HSMM}}\, a_{j,i}\, \mathcal{N}(s|\mu_i^S, \Sigma_i^S) \prod_{c=t-s+1}^{t} \mathcal{N}(\boldsymbol{\xi}_c|\, \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i) \tag{6}$$

where the initialization is given by $\alpha_{1,i}^{\text{HSMM}} = \Pi_i\, \mathcal{N}(1|\mu_i^S, \Sigma_i^S)\, \mathcal{N}(\boldsymbol{\xi}_1|\, \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)$, and the output distribution in state i is conditionally independent for the s duration steps given as $\prod_{c=t-s+1}^{t} \mathcal{N}(\boldsymbol{\xi}_c|\, \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)$. Note that for $t < s^{\max}$, the sum over duration steps is computed for $t - 1$ steps, instead of $s^{\max}$. Without the observation sequence for the next time steps, the forward variable simplifies to

$$\alpha_{t,i}^{\text{HSMM}} = \sum_{s=1}^{\min(s^{\max}, t-1)} \sum_{j=1}^{K} \alpha_{t-s,j}^{\text{HSMM}}\, a_{j,i}\, \mathcal{N}(s|\mu_i^S, \Sigma_i^S) \tag{7}$$

The forward variable is used to plan the movement sequence for the next T_p steps with $t = t + 1 \ldots T_p$. During prediction, we only use the transition matrix and the duration model to plan the future evolution of the initial/current state and omit the influence of the spatial data that we cannot observe, i.e., $\mathcal{N}(\boldsymbol{\xi}_t|\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i) = 1$ for $t > 1$. This is used to retrieve a step-wise reference trajectory $\mathcal{N}(\hat{\boldsymbol{\mu}}_t, \hat{\boldsymbol{\Sigma}}_t)$ from a given state sequence z_t computed from the forward variable with

$$z_t = \{z_t, \ldots, z_{T_p}\} = \arg\max_i \; \alpha_{t,i}^{\text{HSMM}} \tag{8}$$

$$\hat{\boldsymbol{\mu}}_t = \boldsymbol{\mu}_{z_t}, \quad \hat{\boldsymbol{\Sigma}}_t = \boldsymbol{\Sigma}_{z_t} \tag{9}$$

Figure 4 shows a conceptual representation of the step-wise sequence of states generated by deterministically sampling from HSMM encoding of the Z-shaped data. In the next section, we show how to synthesise robot movement from this step-wise sequence of states in a smooth manner.

3.5. Motion generation with LQT

We formulate the motion generation problem given the step-wise desired sequence of states $\{\mathcal{N}(\hat{\boldsymbol{\mu}}_t, \hat{\boldsymbol{\Sigma}}_t)\}_{t=1}^{T_p}$ as sequential optimization of a scalar cost function with LQT (Borrelli et al., 2011). The control policy $\boldsymbol{u}_t \in \mathbb{R}^P$ at each time step is obtained by minimizing the cost function over the *finite time horizon* T_p,

$$c_t(\boldsymbol{\xi}_t, \boldsymbol{u}_t) = \sum_{t=1}^{T_p} (\boldsymbol{\xi}_t - \hat{\boldsymbol{\mu}}_t)^{\text{T}} \boldsymbol{Q}_t (\boldsymbol{\xi}_t - \hat{\boldsymbol{\mu}}_t) + \boldsymbol{u}_t^{\text{T}} \boldsymbol{R}_t \boldsymbol{u}_t \tag{10}$$

$$\text{s.t.} \quad \boldsymbol{\xi}_{t+1} = \boldsymbol{A}_d \boldsymbol{\xi}_t + \boldsymbol{B}_d \boldsymbol{u}_t$$

starting from the initial state $\boldsymbol{\xi}_1$ and following the discrete linear dynamical system specified by $\boldsymbol{A}_d$ and $\boldsymbol{B}_d$ with positive-definite weighting matrices $\boldsymbol{Q}_t$ and $\boldsymbol{R}_t$. We consider a linear time-invariant double integrator system to describe the system dynamics. Alternatively, a time-varying linearization of the system dynamics along the reference trajectory can also be used to model the system dynamics without loss of generality. Both discrete and continuous time linear quadratic regulator/tracker can be used to follow the desired trajectory. The discrete time formulation, however, gives numerically stable results for a wide range of values of $\boldsymbol{R}_t$. $\boldsymbol{Q}_t$ is set inversely proportional to the variance in the observed demonstrated trajectories, $\boldsymbol{Q}_t = \hat{\boldsymbol{\Sigma}}_t^{-1}$, i.e., parts of the demonstrations where variance is higher, the cost weight is lower to produce lower stiffness and damping. The control law $\boldsymbol{u}_t^*$ that minimizes the cost function in (10) under finite horizon subject to the linear dynamics in discrete time is given as

$$\boldsymbol{u}_t^* = \boldsymbol{K}_t(\hat{\boldsymbol{\mu}}_t - \boldsymbol{\xi}_t) + \boldsymbol{u}_t^{\text{FF}} \tag{11}$$

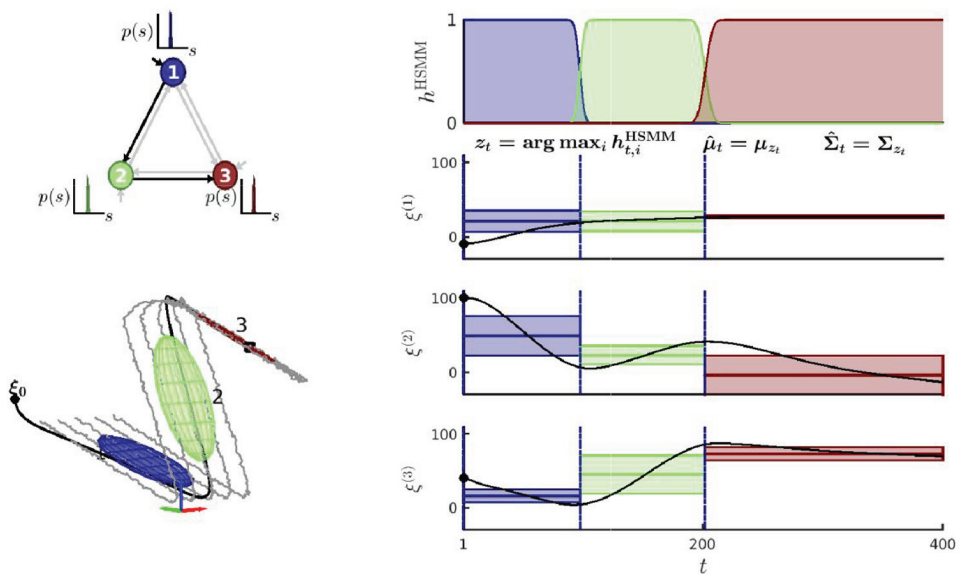


Fig. 4. Sampling from HSMM from an unseen initial state $\boldsymbol{\xi}_0$ over the next time horizon and tracking the step-wise desired sequence of states $\mathcal{N}(\hat{\boldsymbol{\mu}}_t, \hat{\boldsymbol{\Sigma}}_t)$ with an LQT controller. Note that this converges although $\boldsymbol{\xi}_0$ was not previously encountered.

$$\begin{aligned}\boldsymbol{u}_t^* = &- \left(\boldsymbol{R} + \boldsymbol{B}_d^{\mathrm{T}} \boldsymbol{P}_t \boldsymbol{B}_d\right)^{-1} \boldsymbol{B}_d^{\mathrm{T}} \boldsymbol{P}_t \boldsymbol{A}_d (\boldsymbol{\xi}_t - \hat{\boldsymbol{\mu}}_t) \\ &- \left(\boldsymbol{R} + \boldsymbol{B}_d^{\mathrm{T}} \boldsymbol{P}_t \boldsymbol{B}_d\right)^{-1} \boldsymbol{B}_d^{\mathrm{T}} (\boldsymbol{P}_t (\boldsymbol{A}_d \hat{\boldsymbol{\mu}}_t - \hat{\boldsymbol{\mu}}_t) + \boldsymbol{d}_t)\end{aligned}$$

where $\boldsymbol{K}_t = [\boldsymbol{K}_t^{\mathcal{P}}, \boldsymbol{K}_t^{\mathcal{V}}]$ are the full stiffness and damping matrices for the feedback term, and $\boldsymbol{u}_t^{\mathrm{FF}} = \left(\boldsymbol{R} + \boldsymbol{B}_d^{\mathrm{T}} \boldsymbol{P}_t \boldsymbol{B}_d\right)^{-1} \boldsymbol{B}_d^{\mathrm{T}} (\boldsymbol{P}_t (\boldsymbol{A}_d \hat{\boldsymbol{\mu}}_t - \hat{\boldsymbol{\mu}}_t) + \boldsymbol{d}_t)$ is the feedforward term. The stiffness and damping parameters are obtained as a closed-form solution of the linear quadratic regulator with time-varying quadratic cost function, such that the controller in (11) minimizes the total cost accumulated over the finite horizon (see the Appendix for details). Here $\boldsymbol{P}_t$ and $\boldsymbol{d}_t$ are obtained by solving the Riccati differential equation and linear differential equation backwards in discrete time from terminal conditions $\boldsymbol{P}_{T_p} = \boldsymbol{Q}_{T_p}$ and $\boldsymbol{d}_{T_p} = \boldsymbol{0}$, respectively,

$$\begin{aligned}\boldsymbol{P}_{t-1} &= \boldsymbol{Q}_t - \boldsymbol{A}_d^{\mathrm{T}} (\boldsymbol{P}_t \boldsymbol{B}_d \left(\boldsymbol{R} + \boldsymbol{B}_d^{\mathrm{T}} \boldsymbol{P}_t \boldsymbol{B}_d\right)^{-1} \boldsymbol{B}_d^{\mathrm{T}} \boldsymbol{P}_t - \boldsymbol{P}_t) \boldsymbol{A}_d \\ \boldsymbol{d}_{t-1} &= (\boldsymbol{A}_d^{\mathrm{T}} - \boldsymbol{A}_d^{\mathrm{T}} \boldsymbol{P}_t \boldsymbol{B}_d \left(\boldsymbol{R} + \boldsymbol{B}_d^{\mathrm{T}} \boldsymbol{P}_t \boldsymbol{B}_d\right)^{-1} \boldsymbol{B}_d^{\mathrm{T}}) \\ &\quad (\boldsymbol{P}_t \left(\boldsymbol{A}_d \hat{\boldsymbol{\mu}}_t - \hat{\boldsymbol{\mu}}_{t+1}\right) + \boldsymbol{d}_t)\end{aligned} \tag{12}$$

Figure 4 shows the results of applying discrete LQT on the desired step-wise sequence of states sampled from an HSMM encoding of the Z-shaped demonstrations. Note that the gains can be precomputed before simulating the system if the reference trajectory does not change during the reproduction of the task. The resulting trajectory $\boldsymbol{\xi}_t^*$ smoothly tracks the step-wise reference trajectory $\hat{\boldsymbol{\mu}}_t$ and the gains $\boldsymbol{K}_t^{\mathcal{P}}, \boldsymbol{K}_t^{\mathcal{V}}$ locally stabilize $\boldsymbol{\xi}_t$ to track $\boldsymbol{\xi}_t^*$ in accordance with the precision required during the task.

4. Invariant task-parameterized model adaptation

Conventional approaches to encode task variations such as change in the pose of the object is to augment the state of the environment with the policy parameters (Paraschos et al., 2013). Such an encoding, however, does not capture the geometric structure of the problem. In contrast, we exploit the problem structure by introducing the task parameters in the form of coordinate systems that observe the demonstrations from multiple perspectives. Task-parameterization enables the model parameters to adapt in accordance with the external task parameters that describe the environmental situation, instead of hard coding the solution for each new situation or handling it in an *ad hoc* manner (Calinon, 2016; Pervez and Lee, 2018; Tanwani and Calinon, 2016; Wilson and Bobick, 1999). When a different situation occurs (pose of the object changes), changes in the task parameters/reference frames are used to modulate the model parameters in order to adapt the robot movement to the new situation.

4.1. Model learning

We represent the task parameters with F coordinate systems, defined by $\{\boldsymbol{A}_j, \boldsymbol{b}_j\}_{j=1}^F$, where $\boldsymbol{A}_j$ denotes the orientation of the frame as a rotation matrix and $\boldsymbol{b}_j$ represents the origin of the frame. We assume that the coordinate frames are specified by the user, based on prior knowledge about the carried out task depending upon the invariant components in the task. Typically, coordinate frames are attached to objects, tools, or locations that could be relevant in the

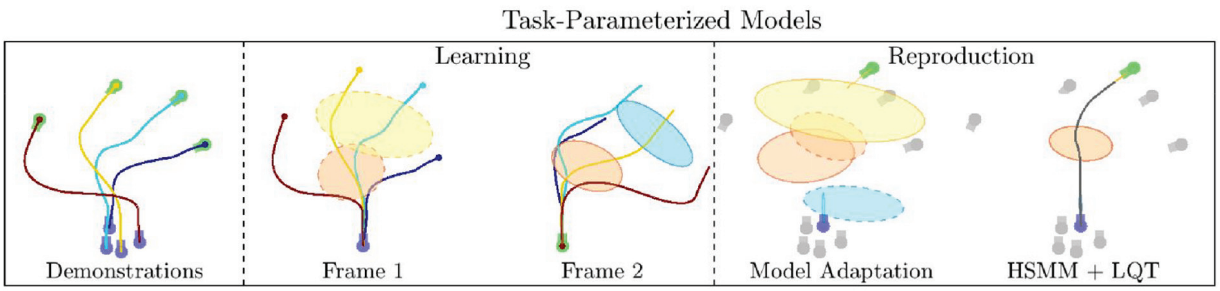


Fig. 5. Task-parameterized formulation of HSMM: four demonstrations on left are observed from two coordinate systems that define the start and end position of the demonstration (starting in purple position and ending in green position for each demonstration). The generative model is learned in the respective coordinate systems. The model parameters in respective coordinate systems are adapted to the new unseen object positions by computing the products of linearly transformed Gaussian mixture components. The resulting HSMM is combined with LQT for smooth retrieval of manipulation tasks. Note that if we add a moving obstacle with respect to which the model should be invariant, then we would need three frames. Similarly, if we are only interested in grasping an object from different initial conditions, we would only need one frame on the object to be grasped.

execution of a task (see Figure 5 and Section 6 for details). Each datapoint $\boldsymbol{\xi}_t$ is observed from the viewpoint of F different experts/frames, with $\boldsymbol{\xi}_t^{(j)} = \boldsymbol{A}_j^{-1}(\boldsymbol{\xi}_t - \boldsymbol{b}_j)$ denoting the datapoint observed with respect to frame j. The parameters of the task-parameterized HSMM are defined by

$$\theta = \{\{\boldsymbol{\mu}_i^{(j)}, \boldsymbol{\Sigma}_i^{(j)}\}_{j=1}^F, \{a_{i,m}\}_{m=1}^K, \mu_i^S, \Sigma_i^S\}_{i=1}^K$$

where $\boldsymbol{\mu}_i^{(j)}$ and $\boldsymbol{\Sigma}_i^{(j)}$ define the mean and the covariance matrix of i th mixture component in frame j. Parameter updates of the task-parameterized HSMM algorithm remain the same as HSMM, except the computation of the mean and the covariance matrix is repeated for each coordinate system separately. The emission distribution of the i th state is represented by the product of the probabilities of the datapoint to belong to the i th Gaussian in the corresponding j th coordinate system. The forward variable of HMM in the task-parameterized formulation is described as

$$\alpha_{t,i}^{\text{TP-HMM}} = \left(\sum_{j=1}^K \alpha_{t-1,j}^{\text{HMM}} a_{j,i}\right) \prod_{j=1}^F \mathcal{N}(\boldsymbol{\xi}_t^{(j)} | \boldsymbol{\mu}_i^{(j)}, \boldsymbol{\Sigma}_i^{(j)}) \quad (13)$$

Similarly, the backward variable $\beta_{t,i}^{\text{TP-HMM}}$, the smoothed node marginal $\gamma_{t,i}^{\text{TP-HMM}}$, and the smoothed edge marginal $\zeta_{t,i,j}^{\text{TP-HMM}}$ can be computed by replacing the emission distribution $\mathcal{N}(\boldsymbol{\xi}_t | \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)$ with the product of probabilities of the datapoint in each frame $\prod_{j=1}^F \mathcal{N}(\boldsymbol{\xi}_t^{(j)} | \boldsymbol{\mu}_i^{(j)}, \boldsymbol{\Sigma}_i^{(j)})$. The duration model $\mathcal{N}(s | \mu_i^S, \Sigma_i^S)$ is used as a replacement of the self-transition probabilities $a_{i,i}$. The hidden state sequence over all demonstrations is used to define the duration model parameters $\{\mu_i^S, \Sigma_i^S\}$ as the mean and the standard deviation of staying s consecutive time steps in the i th state.

4.2. Model adaptation in new situations

In order to combine the output from coordinate frames of reference for an unseen situation during testing represented by the frames $\{\tilde{\boldsymbol{A}}_j, \tilde{\boldsymbol{b}}_j\}_{j=1}^F$, we linearly transform the Gaussians back to the global coordinates with $\{\tilde{\boldsymbol{A}}_j, \tilde{\boldsymbol{b}}_j\}_{j=1}^F$, and retrieve the new model parameters $\{\tilde{\boldsymbol{\mu}}_i, \tilde{\boldsymbol{\Sigma}}_i\}$ for the i th mixture component by computing the products of the linearly transformed Gaussians (see Figure 5)

$$\mathcal{N}(\tilde{\boldsymbol{\mu}}_i, \tilde{\boldsymbol{\Sigma}}_i) \propto \prod_{j=1}^F \mathcal{N}\left(\tilde{\boldsymbol{A}}_j \boldsymbol{\mu}_i^{(j)} + \tilde{\boldsymbol{b}}_j, \tilde{\boldsymbol{A}}_j \boldsymbol{\Sigma}_i^{(j)} \tilde{\boldsymbol{A}}_j^{\mathrm{T}}\right) \quad (14)$$

Evaluating the products of Gaussians represents the observation distribution of HSMM, whose output sequence is decoded and combined with LQT for smooth motion generation as shown in the previous section:

$$\begin{aligned} \tilde{\boldsymbol{\Sigma}}_i &= \left(\sum_{j=1}^F \left(\tilde{\boldsymbol{A}}_j \boldsymbol{\Sigma}_i^{(j)} \tilde{\boldsymbol{A}}_j^{\mathrm{T}}\right)^{-1}\right)^{-1} \\ \tilde{\boldsymbol{\mu}}_i &= \tilde{\boldsymbol{\Sigma}}_i \sum_{j=1}^F \left(\tilde{\boldsymbol{A}}_j \boldsymbol{\Sigma}_i^{(j)} \tilde{\boldsymbol{A}}_j^{\mathrm{T}}\right)^{-1} \left(\tilde{\boldsymbol{A}}_j \boldsymbol{\mu}_i^{(j)} + \tilde{\boldsymbol{b}}_j\right) \end{aligned} \quad (15)$$

Note that we transform the movement relative to the reference frames as being observed from multiple reference frames, and then combine this information with the products of linearly transformed Gaussians. In case there is only one frame of reference for a goal-directed movement, the approach is equivalent to encoding the HSMM with relative positions between the object and the end-effector.

4.3. Latent space representations

Model-based generative models such as HSMMs tend to suffer from the *curse of dimensionality* when few datapoints are available. We use statistical subspace clustering

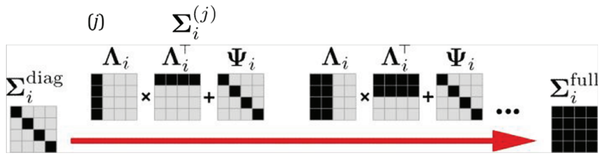


Fig. 6. Parameters representation of a diagonal, full, and MFA decomposition of covariance matrix. Filled blocks represent non-zero entries.

methods that reduce the number of parameters to be robustly estimated to address this problem. A simple way to reduce the number of parameters would be to constrain the covariance structure to a diagonal or spherical/isotropic matrix, and restrict the number of parameters at the cost of treating each dimension separately. Such decoupling, however, cannot encode the important motor control principles of coordination, synergies and action-perception couplings (Wolpert et al., 2011).

Consequently, we seek out a latent feature space in the high-dimensional data to reduce the number of model parameters that can be robustly estimated. We consider three formulations to this end: (1) low-rank decomposition of the covariance matrix using *mixture of factor analyzers* (MFA) approach (McLachlan et al., 2003); (2) partial tying of the covariance matrices of the mixture model with the same set of basis vectors, albeit different scale with semi-tied covariance matrices (Gales, 1999; Tanwani and Calinon, 2016); and (3) Bayesian non-parametric sequence clustering under small variance asymptotics (SVA) (Kulis and Jordan, 2012; Roychowdhury et al., 2013; Tanwani and Calinon, 2019). All the decompositions can readily be combined with invariant task-parameterized HSMM and LQT for encapsulating reactive autonomous behavior as shown in the previous section.

4.3.1. MFA. The basic idea of MFA is to perform subspace clustering by assuming the covariance structure for each component of the form,

$$\boldsymbol{\Sigma}_i = \boldsymbol{\Lambda}_i \boldsymbol{\Lambda}_i^{\mathrm{T}} + \boldsymbol{\Psi}_i \tag{16}$$

where $\boldsymbol{\Lambda}_i \in \mathbb{R}^{D \times d}$ is the *factor loadings matrix* with $d < D$ for parsimonious representation of the data, and $\boldsymbol{\Psi}_i$ is the diagonal noise matrix (see Figure 6 for MFA representation in comparison to a diagonal and a full covariance matrix). Note that the mixture of probabilistic principal component analysis (MPPCA) model is a special case of MFA with the distribution of the errors assumed to be isotropic with $\boldsymbol{\Psi}_i = \boldsymbol{I}\sigma_i^2$ (Tipping and Bishop, 1999).

The MFA model assumes that $\boldsymbol{\xi}_t$ is generated using a linear transformation of d-dimensional vector of latent (unobserved) factors $\boldsymbol{f}_t$,

$$\boldsymbol{\xi}_t = \boldsymbol{\Lambda}_i \boldsymbol{f}_t + \boldsymbol{\mu}_i + \boldsymbol{\epsilon} \tag{17}$$

where $\boldsymbol{\mu}_i \in \mathbb{R}^D$ is the mean vector of the i th factor analyzer, $\boldsymbol{f}_t \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$ is a normally distributed factor, and $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Psi}_i)$ is a zero-mean Gaussian noise with diagonal covariance $\boldsymbol{\Psi}_i$. The diagonal assumption implies that the observed variables are independent given the factors.

Note that the subspace of each cluster is not spanned by orthogonal vectors with MFA, whereas it is a necessary condition in models based on eigendecomposition such as principal component analysis (PCA). Each covariance matrix of the mixture component has its own subspace spanned by the basis vectors of $\boldsymbol{\Sigma}_i$. As the number of components increases to encode more complex skills, an increasing large number of potentially redundant parameters are used to fit the data. Consequently, there is a need to share the basis vectors across the mixture components such as semi-tying the covariance matrices of the mixture model.

4.3.2. Semi-tied mixture model. When the covariance matrices of the mixture model share the same set of parameters for the latent feature space, we call the model a *semi-tied* mixture model (Tanwani and Calinon, 2016). The main idea behind semi-tied mixture models is to decompose the covariance matrix $\boldsymbol{\Sigma}_i$ into two terms: a common latent feature matrix $\boldsymbol{H} \in \mathbb{R}^{D \times D}$ and a component-specific diagonal matrix $\boldsymbol{\Sigma}_i^{(\text{diag})} \in \mathbb{R}^{D \times D}$, i.e.,

$$\boldsymbol{\Sigma}_i = \boldsymbol{H} \boldsymbol{\Sigma}_i^{(\text{diag})} \boldsymbol{H}^{\mathrm{T}} \tag{18}$$

The latent feature matrix encodes the locally important synergistic directions represented by D non-orthogonal basis vectors that are shared across all the mixture components, whereas the diagonal matrix selects the appropriate subspace of each mixture component as convex combination of a subset of the basis vectors of $\boldsymbol{H}$. Note that the eigendecomposition of $\boldsymbol{\Sigma}_i = \boldsymbol{U}_i \boldsymbol{\Sigma}_i^{(\text{diag})} \boldsymbol{U}_i^{\mathrm{T}}$ contains D basis vectors of $\boldsymbol{\Sigma}_i$ in $\boldsymbol{U}_i$. In comparison, semi-tied mixture model gives D globally representative basis vectors that are shared across all the mixture components. The parameters $\boldsymbol{H}$ and $\boldsymbol{\Sigma}_i^{(\text{diag})}$ are updated in closed form with EM updates of HSMM (Gales, 1999).

The underlying hypothesis in semi-tying the model parameters is that similar coordination patterns occur at different phases in a manipulation task. By exploiting the spatial and temporal correlation in the demonstrations, we reduce the number of parameters to be estimated while locking the most important synergies to cope with perturbations. This allows the reuse of the discovered synergies in different parts of the task having similar coordination patterns. In contrast, the MFA decomposition of each covariance matrix separately cannot exploit the temporal synergies, but has more flexibility in locally encoding the data.

The encoding of Z-shaped demonstrations with semi-tied model in Figure 7 reveals the locally important basis vectors comprising the latent feature space $\boldsymbol{H}$. In contrast,

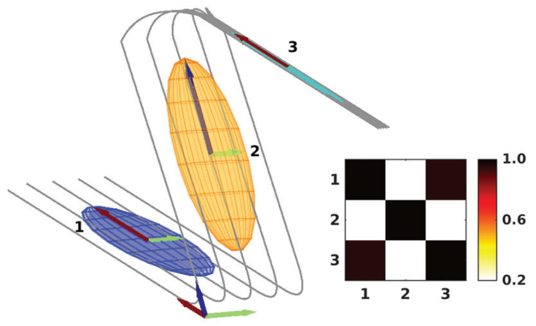


Fig. 7. Left: Semi-tied mixture model encoding of Z-shaped data with 3 components and basis vectors of $\boldsymbol{H}$ shown at the origin. Right: Pairwise correlation among the mixture components of semi-tied mixture model.

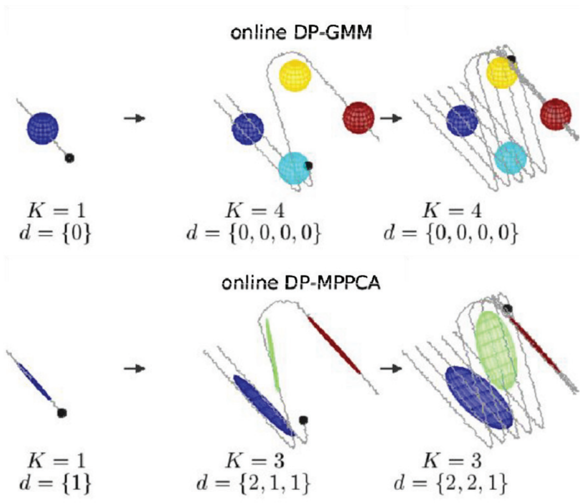


Fig. 8. Bayesian non-parametric clustering of Z-shaped streaming data under SVA with: (top) online DP-GMM; (bottom) online DP-MPPCA. Note that the number of clusters and the subspace dimension of each cluster is adapted in a non-parametric manner.

PCA here would yield orthogonal basis vectors along the directions of largest variance globally. Note that the basis vectors are not required to be orthogonal in the semi-tied GMM. It can be seen in Figure 7 that the basis vector in red is shared across the first and the third mixture component, while the basis vector in green is shared across the first and the second mixture component. The basis vector in blue is tied only to the second mixture component. This yields high correlation between the first and the third mixture component, and low correlation of the second Gaussian component with other mixture components (see Figure 7(right)).

4.3.3. Bayesian non-parametrics under SVA. Specifying the number of latent states in a mixture model is often difficult. Model selection methods such as cross-validation or Bayesian information criterion (BIC) are typically used to determine the number of states. Bayesian non-parametric approaches comprising hierarchical Dirichlet processes (HDPs) provide a principled model selection procedure by Bayesian inference in an HMM with infinite number of states (Teh et al., 2006). These approaches provide flexibility in model selection, however, their widespread use is limited by the computational overhead of existing sampling-based and variational techniques for inference. We take a *SVA* approximation of the Bayesian non-parametric model that collapses the posterior to a simple deterministic model, while retaining the non-parametric characteristics of the algorithm.

SVA analysis implies that the covariance matrix $\boldsymbol{\Sigma}_i$ of all the Gaussians is set to the isotropic noise σ^2, i.e., $\boldsymbol{\Sigma}_i \approx \lim_{\sigma^2\to 0}\sigma^2\boldsymbol{I}$ in the likelihood function and the prior distribution (Broderick et al., 2013; Kulis and Jordan, 2012). The analysis yields simple deterministic models, while retaining the non-parametric nature. For example, SVA analysis of the Bayesian non-parametric GMM leads to the DP-means algorithm (Kulis and Jordan, 2012). Similarly, SVA analysis of the Bayesian non-parametric HMM under HDP yields the segmental k-means problem (Roychowdhury et al., 2013).

Restricting the covariance matrix to an isotropic/spherical noise, however, fails to encode the coordination patterns in the demonstrations. Consequently, we model the covariance matrix in its intrinsic affine subspace of dimension d_i with projection matrix $\boldsymbol{\Lambda}_i^{d_i} \in \mathbb{R}^{D\times d_i}$, such that $d_i < D$ and $\boldsymbol{\Sigma}_i = \lim_{\sigma^2\to 0}\boldsymbol{\Lambda}_i^{d_i}\boldsymbol{\Lambda}_i^{d_i \mathsf{T}} + \sigma^2\boldsymbol{I}$ (akin to a mixture of probabilistic PCA Tipping and Bishop (1999)). Under this assumption, we apply the SVA limit on the remaining $(D-d_i)$ dimensions to encode the most important coordination patterns while being parsimonious in the number of parameters (see Figure 8). Performing SVA of the joint likelihood of HDP–HMM yields the maximum *a posteriori* estimates of the parameters by iteratively minimizing the loss function[1]

$$\mathcal{L}(\boldsymbol{z},\boldsymbol{d},\boldsymbol{\mu},\boldsymbol{U},\boldsymbol{a}) = \sum_{t=1}^{T}\operatorname{dist}(\boldsymbol{\xi}_t,\boldsymbol{\mu}_{z_t},\boldsymbol{U}_{z_t}^{d_i})^2 + \lambda(K-1)$$
$$+\lambda_1\sum_{i=1}^{K} d_i - \lambda_2\sum_{t=1}^{T-1}\log(a_{z_t,z_{t+1}}) + \lambda_3\sum_{i=1}^{K}(\tau_i - 1)$$

where $\operatorname{dist}(\boldsymbol{\xi}_t,\boldsymbol{\mu}_{z_t},\boldsymbol{U}_{z_t}^{d})^2$ represents the distance of the datapoint $\boldsymbol{\xi}_t$ to the subspace of cluster z_t defined by mean $\boldsymbol{\mu}_{z_t}$ and unit eigenvectors of the covariance matrix $\boldsymbol{U}_{z_t}^{d}$ (see the supplementary material for details). The algorithm optimizes the number of clusters and the subspace dimension of each cluster while minimizing the distance of the datapoints to the respective subspaces of each cluster. The λ_2 term favors the transitions to states with higher transition probability (states which have been visited more often before), λ_3 penalizes for transition to unvisited states with

τ_i denoting the number of distinct transitions out of state i, while λ and λ_1 are the penalty terms for increasing the number of states and the subspace dimension of each output state distribution.

The analysis is used here for scalable online sequence clustering that is non-parametric in the number of clusters and the subspace dimension of each cluster. The resulting algorithm groups the data in its low-dimensional subspace with non-parametric mixture of probabilistic principal component analyzers based on the Dirichlet process, and captures the state transition and state duration information in a HDP–HSMM. The cluster assignment and the parameter updates at each iteration minimize the loss function, thereby, increasing the model fitness while penalizing for new transitions, new dimensions, and/or new clusters. The interested reader can find more details of the algorithm in Tanwani and Calinon (2019). We further illustrate the importance of latent space models in efficiently encoding the observations in the experiments section.

5. Imitation learning from videos

Learning meaningful visual representations in an embedding space can facilitate generalization in downstream tasks such as action segmentation and imitation (Tanwani et al., 2020). To make it concrete, consider $\{\boldsymbol{I}_{n,t}\}_{n=1,t=1}^{N,T_n}$ as a set of N video demonstrations, where $\boldsymbol{I}_{n,t} \in \mathbb{R}^{640\times 480\times 3}$ denotes the RGB image at time t of the n th demonstration comprising T_n datapoints. Each demonstration describes a video of a manipulation skill such as pick-and-place or surgical suturing task, collected from a third-person viewpoint that does not change in a demonstration but may change across demonstrations. We assume access to a supervisor that assigns segment labels to each frame as belonging to one of the segments $z_{n,t} \in \{1\dots C\}$ such as reach and grasp, resulting in a set of labeled demonstrations $\{\boldsymbol{I}_{n,t}, z_{n,t}\}_{n=1,t=1}^{N,T_n}$. Without loss of generality, we drop the index n to denote an image frame as $\boldsymbol{I}_t$ for the rest of the paper.

We seek to learn a deep motion-centric representation of video observations $f_{\boldsymbol{\Theta}_\text{D}} : \boldsymbol{I}_t \to \boldsymbol{\xi}_t$ with $\boldsymbol{\xi}_t \in \mathbb{R}^d$ and $d \ll |\boldsymbol{I}_t|$ such that similar action segments are grouped together in the embedding space while being invariant to the nuisance variables such as lighting, background, and camera viewpoint. The representation needs to semantically align the videos in the embedding space, and subsequently predict segment labels $z_{t-l:t}$ for the training mini-batch of length l with a sequence learning model $h_{\boldsymbol{\Theta}_\text{S}} : \boldsymbol{\xi}_{t-l:t} \to z_{t-l:t}$, i.e.,

$$\boldsymbol{\xi}_t = f(\boldsymbol{I}_t; \boldsymbol{\Theta}_\text{D}) \tag{19}$$

$$z_{t-l:t} = h(\boldsymbol{\xi}_{t-l:t}; \boldsymbol{\Theta}_\text{D}, \boldsymbol{\Theta}_\text{S}) \tag{20}$$

The learned networks are subsequently used to train the control policy in the embedding space $\pi_{\boldsymbol{\Theta}_R} : \boldsymbol{\xi}_t \to \boldsymbol{u}_t$, where $\boldsymbol{u}_t \in \mathbb{R}^p$ corresponds to the end-effector pose of the robot arm to imitate the manipulation skill in the video demonstrations. In this work, we assume access to the kinematic poses of the end-effector in the video demonstrations, and defer autonomous learning of the manipulation skill from the embedded observations to the future work.

We follow a three-step methodology for imitation learning from videos (see Figure 2): (1) learn the deep embedding space to pull together similar segments close while pushing away other far segments with metric learning; (2) train the sequence model parameters using the embedded observations; and (3) learn the control policy from the embedded observations to control the end-effector pose of the robot arm.

5.1. Deep embedding space

The deep embedding space reflects the task-relevant attributes of the objects in the videos and how they can be mapped onto the robot end-effector. Sample and time complexity of collecting/labeling videos at pixel level to reflect such associations can be very high for training vision-based deep models in robotics. The trajectory-centric invariant formulations, described in the previous section, may not readily generalize to the video demonstrations as: (1) transforming the images with respect to an arbitrary viewpoint is non-trivial as it requires the full 3D reconstruction of the environment, (2) the variance across images per pixel may not be indicative of the representative features in the demonstrations.

Here, we learn a deep embedding representation using a metric learning loss which pulls together observations from the same action segment in the embedding space, while pushing away observations from other action segments that functionally correspond to different sub-goals or movement primitives. We use triplet loss for metric learning in this work (Schroff et al., 2015). Note that the contrastive loss or the magnet loss may also be used in a similar way (Rippel et al., 2015). During training, the loss operates on the tuple corresponding to the anchor image embedding $\boldsymbol{\xi}_t$, a positive sample belonging to the same action segment $\boldsymbol{\xi}_t^+$ and a negative sample randomly chosen from another action segment $\boldsymbol{\xi}_t^-$. Triplet loss posits that the distance of the anchor to the positive sample in the embedding space is less than the distance to the negative sample by some constant margin ζ, i.e.,

$$\mathcal{L}(\boldsymbol{\Theta}_\text{D}) = \frac{1}{T}\sum_{t=1}^{T}\{\| \boldsymbol{\xi}_t - \boldsymbol{\xi}_t^+ \|_2^2 - \| \boldsymbol{\xi}_t - \boldsymbol{\xi}_t^- \|_2^2 + \zeta\}_+ \tag{21}$$

where $\{.\}_+$ is the hinge loss and the representation $\boldsymbol{\xi}_t$ is normalized to extract scale-invariant features similar to (Schroff et al., 2015). We compare the triplet metric learning loss with other embedding approaches including: (1) *incremental principal component analysis* (iPCA) to project video observations into an uncorrelated embedding space (Zhao et al., 2006); (2) TCC to align the embedding space by matching frames over time across video demonstrations (Dwibedi et al., 2019); (3) *time-contrastive network* with single view (svTCN) using a window of six neighboring frames in the sequence to find the positive sample for each anchor image and negative sample from a window of 12 neighboring frames (Sermanet et al., 2017); (4) *n-pairs* metric that takes pairs of images from same segment labels where a pair is used as an anchor and a positive image, respectively, while each pair in the mini-batch may have different labels. The *n*-pairs loss repels a positive sample from all negative samples in comparison with the nearest negative sample in triplet loss (Sohn, 2016). Note that iPCA, TCC, and svTCN are used in an unsupervised way, whereas we use triplet loss and *n*-pairs loss in a supervised manner with labeled action segments.

5.2. Sequence learning of action segments

We seek to capture the spatiotemporal dependencies in the embedded observations and predict action segments with a sequence learning model. We use an RNN to discriminatively model the action assignment to the observation sequence $P(z_{t-l:t}|\boldsymbol{\xi}_{t-l:t})$ using a stride of length l in a mini-batch. An RNN maintains an additional hidden state and uses the previous hidden state and the current input $\boldsymbol{\xi}_t$ to produce a new hidden state and the output z_t. The hidden state preserves the effect of previous observations in predicting the current output. We use the bi-directional long short-term memory (LSTM) (Hochreiter and Schmidhuber, 1997) in this work that also preserves the effect of future observations within a sequence. We minimize the cross-entropy loss between the true and the predicted labels during training with backpropogration through time (Graves, 2012).

We compare RNNs with *k -nearest neighbor* (KNN) classification accuracy in the embedding space, along with other sequence learning models, namely: (1) CRFs that encode the conditional probability distribution of the output labels sequence given the input observation sequence in a discriminative manner (Lafferty et al., 2001; Sutton and McCallum, 2012; Vakanski et al., 2012); (2) HMMs; (3) HSMMs. Note that CRFs and RNNs are discriminative models trained in a supervised manner, whereas HMM and HSMMs are generative models used in an unsupervised way.

5.3. Imitating end-effector poses

Given the learned model parameters of the embedding space, we map the embedded observation $\boldsymbol{\xi}_t$ to the end-effector pose $\boldsymbol{u}_t$ of the robot arm with a feedforward neural network $\pi_{\boldsymbol{\Theta}_R}$. The feedforward network is defined on top of a pretrained embedding network whose parameters are frozen during the learning process. The pose imitation loss is a weighted combination of the position loss measured in terms of the mean-squared error between the ground-truth and the predicted end-effector position, and the orientation loss measured in terms of the cosine distance between the ground-truth and the predicted end-effector orientation in quaternion space.

6. Experiments, results, and discussion

In this section, we report the experiments for our approach on: (1) pick-and-place task with the Baxter robot from kinesthetic demonstrations; (2) suturing task with the da Vinci robot from videos publicly available in the JIGSAWS dataset (Gao et al., 2014). Note that we do not model contact dynamics with the needle and the suturing phantom, and only imitate the suturing motions on the kinematic level. We empirically investigate: (1) what metric/sequence learning representations generalize better in terms of the segmentation accuracy; (2) the usefulness of the learned embeddings in imitating the end-effector poses on the da Vinci arms.

6.1. Pick-and-place with obstacle avoidance

Pick-and-place is a standard benchmark in robotics because it can be applied to a wide range of environments and applications. The objective of the task is to place the object in a desired target position by picking it from different initial poses of the object, while adapting the movement to avoid the obstacle. The setup of pick-and-place task with obstacle avoidance is shown in Figure 1. The Baxter robot is required to grasp the glass plate with a suction lever placed in an initial configuration as marked on the setup. The obstacle rod can be vertically displaced to one of the eight target configurations. We describe the task with two frames, one frame for the object initial configuration with $\{\boldsymbol{A}_1, \boldsymbol{b}_1\}$ and other frame for the obstacle $\{\boldsymbol{A}_2, \boldsymbol{b}_2\}$ with $\boldsymbol{A}_2 = \boldsymbol{I}$ and $\boldsymbol{b}_2$ to specify the centre of the obstacle. We collect eight kinesthetic demonstrations with different initial configurations of the object and the obstacle successively displaced upwards as marked with the visual circular tags in the figure. Alternate demonstrations are used for the training set, while the rest are used for the test set. Each observation comprises the end-effector Cartesian position, quaternion orientation, gripper status (open/closed), linear velocity, quaternion derivative, and gripper status derivative with $D = 16$, $F = 2$, and a total of 200 datapoints per demonstration. We represent the frame $\{\boldsymbol{A}_1, \boldsymbol{b}_1\}$ as

$$\boldsymbol{A}_1^{(n)} = \begin{bmatrix} \boldsymbol{R}_1^{(n)} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & 0 \\ \boldsymbol{0} & \mathcal{E}_1^{(n)} & \boldsymbol{0} & \boldsymbol{0} & 0 \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{R}_1^{(n)} & \boldsymbol{0} & 0 \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \mathcal{E}_1(n) & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \quad \boldsymbol{b}_1^{(n)} = \begin{bmatrix} \boldsymbol{p}_1^{(n)} \\ \boldsymbol{0} \\ \boldsymbol{0} \\ \boldsymbol{0} \\ 1 \end{bmatrix} \tag{22}$$

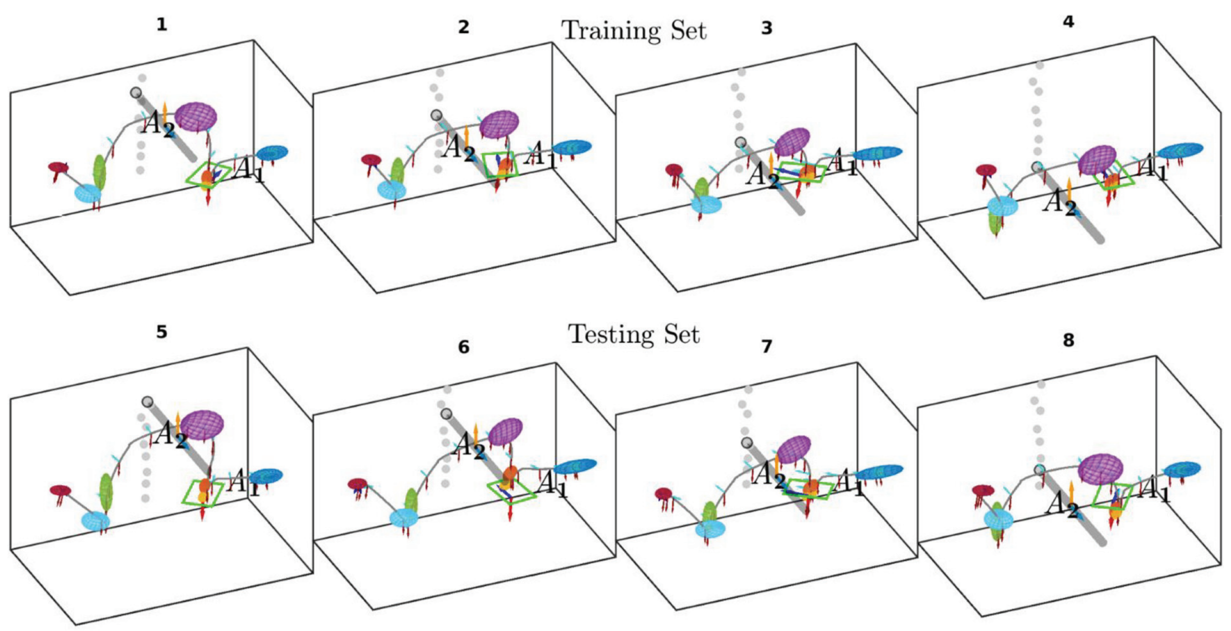


Fig. 9. Task-parameterized HSMM performance on pick-and-place with obstacle avoidance task: (top) training set reproductions; (bottom) testing set reproductions.

Table 1. Performance analysis of invariant HMMs with training MSE, testing MSE, and number of parameters for the pick-and-place task. MSE (in meters) is computed between the demonstrated trajectories and the generated trajectories (lower is better). Latent space formulations give comparable task performance with much fewer parameters.

Model	Training MSE	Testing MSE	Number of parameters
Pick-and-place via obstacle avoidance ($K=7,\ F=2,\ D=16$)			
HSMM	**0.0026±0.0009**	0.014±0.0085	2,198
Semi-tied HSMM	0.0033±0.0016	0.0131±0.0077	1,030
MFA HSMM ($d_k=1$)	0.0037±0.0011	**0.0109±0.0068**	**742**
MFA HSMM ($d_k=4$)	0.0025±0.0007	0.0119±0.0077	1,414
MFA HSMM ($d_k=7$)	0.0023±0.0009	0.0123±0.0084	2,086
SVA HDP HSMM ($K=8, \bar{d}_k=3.94$)	0.0073±0.0024	0.0149±0.0072	1,352

where $\boldsymbol{p}_1^{(n)} \in \mathbb{R}^3$, $\boldsymbol{R}_1^{(n)} \in \mathbb{R}^{3\times3}$, and $\mathcal{E}_1^{(n)} \in \mathbb{R}^{4\times4}$ denote the Cartesian position, the rotation matrix, and the quaternion matrix in the n th demonstration, respectively. Note that we do not consider time as an explicit variable as the duration model in HSMM encapsulates the timing information locally.

Performance setting in our experiments is as follows: $\{\pi_i, \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i\}_{i=1}^K$ are initialized using a k-means clustering algorithm, $\boldsymbol{R}=9\boldsymbol{I}$, where $\boldsymbol{I}$ is the identity matrix, learning converges when the difference of log-likelihood between successive demonstrations is less than 1×10^{-4}. Results of regenerating the movements with seven mixture components are shown in Figure 9. For a given initial configuration of the object, the model parameters are adapted by evaluating the product of Gaussians for a new frame configuration. The reference trajectory is then computed from the initial position of the robot arm using the forward variable of HSMM and tracked using LQT. The robot arm moves from its initial configuration to align itself with the first frame $\{\boldsymbol{A}_1, \boldsymbol{b}_1\}$ to grasp the object, and follows it with the movement to avoid the obstacle and subsequently, align with the second frame $\{\boldsymbol{A}_2, \boldsymbol{b}_2\}$ before placing the object and returning to a neutral position. The model exploits variability in the observed demonstrations to statistically encode different phases of the task such as reach, grasp, move, place, and return. The imposed structure with task parameters and HSMM allows us to acquire a new task in a few human demonstrations, and generalize effectively in picking and placing the object.

Table 1 evaluates the performance of the invariant task-parameterized HSMM with latent space representations. We observe significant reduction in the model parameters with latent space models, while achieving better or similar performance on the unseen situations compared to the task-parameterized HSMM with full covariance matrices. Note that SVA HDP HSMM has similar accuracy with the

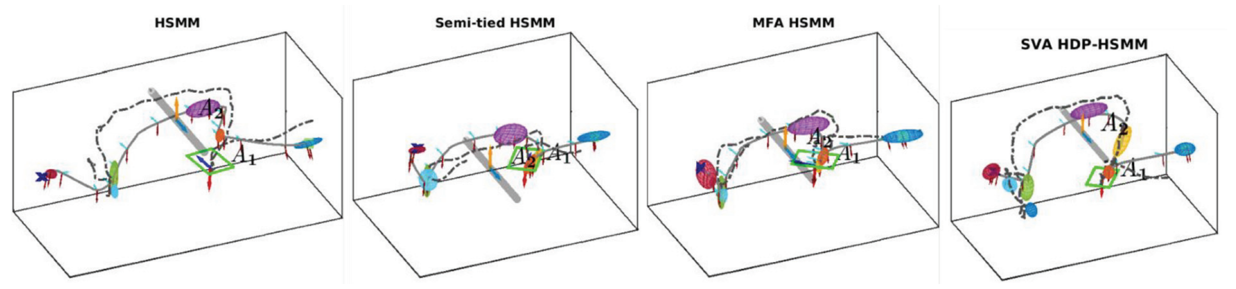


Fig. 10. Latent space representations of invariant task-parameterized HSMM for a randomly chosen demonstration from the test set. Black dotted lines show human demonstration, whereas the gray line shows the reproduction from the model.

advantage that the number of hidden states and the subspace dimensionality of each state cluster is learned from the demonstrations. It is seen that the MFA decomposition gives the best performance on test set with much fewer parameters. A qualitative comparison across models in Figure 10 reveals that the parsimonious representation of latent space models helps to avoid overfitting of skewed Gaussian distributions.

6.2. Surgical suturing

Surgical suturing automation has been studied in several contexts such as needle path planning (Sen et al., 2016), collaborative human–robot suturing (Padoy and Hager, 2011), and learning from demonstrations by trajectory transfer via non-rigid registration in simulation (Schulman et al., 2013). The suturing motions are decomposable into simpler sub-tasks or surgemes that can be inferred from demonstrations (Krishnan et al., 2018; Lea et al., 2015; Murali et al., 2016). In this work, we show how we segment a complex multi-step task into meaningful sub-tasks from surgical videos in order to imitate the intended behavior from videos.

6.3. JIGSAWS dataset

The JIGSAWS dataset contains video demonstrations of three surgical tasks, namely suturing, needle-passing, and knot-tying. We only present results for imitating surgical suturing motions in this work. The suturing dataset consists of eight surgeons with varying skill levels performing the suturing demonstrations five times each on the dual-arm da Vinci robot. Each demonstration consists of a pair of videos from the stereo cameras, kinematic data of the end-effector of the robot arms, and the action segment label for each video frame among a distinct set of 11 suturing sub-tasks as annotated by the experts. The discrete labels correspond to no activity stage `[IDLE]`, reach needle with right hand `[REACH-N-R]`, position needle `[POS-N]`, push needle through tissue `[PUSH-N-T]`, transfer needle from left to right `[TRANS-L-R]`, move to center `[MOVE-C]`, pulling suture with left hand `[PULL-L]`, orienting needle `[ORIENT-N]`, tighten suture with right hand `[TIGHT-R]`, loosening suture `[LOOSE-S]`, and dropping suture at the end `[DROP-S]`. The viewpoint of the camera, lighting and background is fixed in a demonstration, but changes slightly across demonstrations. The suturing style, however, is significantly different across each surgeon. We use a total of 78 demonstrations from the suturing dataset downsampled at 3 frames per second (fps) with an average duration of 3 minutes per video. A total of 62 demonstrations with 4 randomly chosen demonstrations from each surgeon are used for the training set (1 demonstration from a surgeon is corrupted and not used for training), while the remaining demonstration from all surgeons are used as the test set for a total of 16 demonstrations.

6.4. Network architecture(s)

The Siamese network takes as input a downsampled three-channel RGB 320×240 image. The network is augmented on top of the Inception architecture, pre-trained on the ImageNet dataset. We add two convolutional layers of depth 512 each on top of “Mixed-5 d” layer followed by a spatial softmax layer (Finn et al., 2015), a fully connected layer of 2,048 neurons and an embedding layer of 32 dimensions. We use the same Siamese network architecture in all the experiments. This embedding is trained on the triplet loss with a margin of $\zeta = 0.2$. We use a batch size of 128 and 64 for Siamese network and RNN, respectively. Note that 64 batch size at 3 fps corresponds to online segmentation window of 21.3 seconds.

The embedding sequence is fed to a one-layer bi-directional LSTM of 256 hidden neurons. The CRFs network uses 32 potential functions. The HMM/HSMMs are trained in an iterative manner with $K = 30$ hidden states and a multivariate Gaussian in the observation distribution by pooling all the data from the embedding layer after every 1,000 iterations. The number of hidden states are empirically chosen between 1 and 50 components to get the best classification accuracy on the training set. The feedforward network for pose imitation consists of six hidden units with 512, 256, 128, 64, 32, and 16 neurons. The output of policy network corresponds to the

Table 2. Segmentation accuracy performance comparison on the evaluation set averaged over four iterations. Rows correspond to a different embedding space approach, columns correspond to a different segmentation method. KNN results are on training with a Siamese network only. CRF, RNN, *n*-pairs and triplet models are trained in a supervised manner, whereas KNN, HMM, HSMM, PCA, TCC, and svTCN are unsupervised. Motion2Vec (M2V) uses triplet loss with RNN, whereas M2V-T combines the triplet and svTCN loss for temporal alignment. Results are averaged across five trials.

	KNN	HMM	HSMM	CRF	RNN
iPCA	0.586	0.395	0.392	0.415	0.721
TCC	0.667	0.662	0.642	0.601	0.727
svTCN	0.792	0.676	0.661	0.723	0.812
Images	0.748	0.716	0.712	0.811	0.835
***n*-pairs**	0.835	0.799	0.794	0.824	0.854
M2V	0.829	0.831	0.812	0.838	**0.855**
M2V-T	**0.844**	0.828	0.822	0.801	0.843

three-dimensional Cartesian position of the end-effector, four-dimensional quaternion orientation of the end-effector, and a jaw angle for each of the two arms.

6.5. Role of metric and sequence learning in segmentation

Table 2 summarizes the performance comparison of Motion2Vec with different combinations of supervised and unsupervised approaches to metric and sequence learning. We use segmentation accuracy on the test set as the performance metric defined by the percentage of correct segment predictions in comparison with the ground-truth segments annotated by human experts. We observe that svTCN performs better among the unsupervised metric learning approaches without using any of the action segment labels. The unsupervised metric learning approaches perform well with RNN, but other sequence models including HMMs/HSMMs and CRFs find it difficult to encode the action segments. On the other hand, Motion2Vec representation with triplet loss performs well with both the supervised and the unsupervised sequence learning approaches by better grouping the action segments with the use of labeled demonstrations. Triplet loss with RNN gives better performance among all the compared approaches. M2V-T that combines supervised triplet loss with unsupervised time contrastive loss better aligns the images temporally with nearest-neighbor imitation accuracy of 84.4% in the embedding space.

Figure 11 gives a qualitative performance analysis of different sequence learning approaches on a suturing demonstration from the test set. The sequence learning models are able to predict temporally robust action segments on Motion2Vec trained embeddings. Although the video frames in the test set are not observed, Motion2Vec is able to associate them with complex segments such as needle insertion and extraction, while being invariant to the dataset variations including camera viewpoint, background and skill level of the surgeons in the observations.

Figure 12 shows the confusion matrix instance of Motion2Vec with an overall evaluation segmentation accuracy of 86.07%. We observe that the similar neighboring segments in the test set tend to be more often confused suggesting that the related activities are closely grouped in the embedding space.

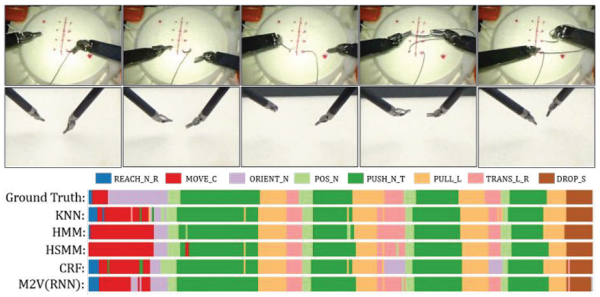


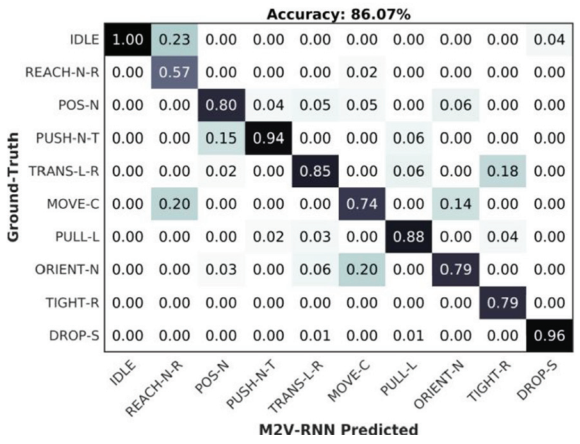
Fig. 11. Top: Qualitative performance comparison of different segmentation policies on top of Motion2Vec with the ground-truth sequence for surgical suturing. Test set image sequence for the first suture is shown in the top, whereas the imitation sequence on the da Vinci robot arms is shown in the bottom. KNN, HMM, and HSMM are unsupervised, whereas CRF and RNN are supervised approaches. In comparison with KNN, RNN gives temporally consistent segments by using the sequential information in the embedded vectors.

6.6. Kinematic imitation on the da Vinci robot

We investigate two scenarios with Motion2Vec embeddings: (1) a single-pose imitation model for all surgeons; (2) a separate pose imitation model for each surgeon. Results of pose imitation from Motion2Vec in comparison to decoding from raw videos are summarized in Table 3. We obtain comparable performance on the test set for all surgeons with raw videos and Motion2Vec embeddings, whereas the per-surgeon pose-decoding model with Motion2Vec gives better performance with position error of 0.94 cm per observation on the test set. We further test the robustness of the embeddings by adding a Gaussian noise of variance 0.15 on top of preprocessed images and observe that Motion2Vec robustly preserves the

Table 3. Pose imitation error in terms of median cosine quaternion loss and root mean squared error (RMSE) in Cartesian three-dimensional position space in centimeters on the test set. Rows indicate pose imitation of: all surgeons from raw videos (images all); all surgeons on Motion2Vec (M2V all); per surgeon pose decoding on M2V (M2V per). Motion2Vec robustly preserves the spatiotemporal alignment in the observations.

	Noise	Median cosine Quat loss	RMSE Position
Images all	–	63.87	1.04
	0.15	89.97	1.50
M2V all	–	61.49	1.15
	0.15	59.89	1.34
M2V per	–	**36.49**	**0.94**
	0.15	**37.21**	**1.12**



	IDLE	REACH-N-R	POS-N	PUSH-N-T	TRANS-L-R	MOVE-C	PULL-L	ORIENT-N	TIGHT-R	DROP-S
IDLE	1.00	0.23	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.04
REACH-N-R	0.00	0.57	0.00	0.00	0.00	0.02	0.00	0.00	0.00	0.00
POS-N	0.00	0.00	0.80	0.04	0.05	0.05	0.00	0.06	0.00	0.00
PUSH-N-T	0.00	0.00	0.15	0.94	0.00	0.00	0.06	0.00	0.00	0.00
TRANS-L-R	0.00	0.00	0.02	0.00	0.85	0.00	0.06	0.00	0.18	0.00
MOVE-C	0.00	0.20	0.00	0.00	0.00	0.74	0.00	0.14	0.00	0.00
PULL-L	0.00	0.00	0.00	0.02	0.03	0.00	0.88	0.00	0.04	0.00
ORIENT-N	0.00	0.00	0.03	0.00	0.06	0.20	0.00	0.79	0.00	0.00
TIGHT-R	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.79	0.00
DROP-S	0.00	0.00	0.00	0.00	0.01	0.00	0.01	0.00	0.00	0.96

Fig. 12. Normalized confusion matrix of Motion2Vec with RNN on the suturing test set, suggesting similar action segments are grouped together in the embedding space. Note that the `[LOOSE-S]` segment was not found in the test set and has been omitted from the plot.

spatiotemporal alignment of the videos. Figure 13 shows the mean of decoded positions (with and without noise) from the embedded observations in comparison to the ground-truth and the raw videos decoded positions. We encourage the readers to see supplementary material video for kinematic imitation in simulation and on real on da Vinci robot arms.

7. Conclusions

Sequential robot imitation learning from demonstrations is a promising approach to teach manipulation skills to robots. We presented invariant task representations with HSMMs for recognition, prediction, and reproduction of trajectory-centric data; along with learning in latent space representations for robust parameter estimation of mixture model. By sampling the sequence of states from the model

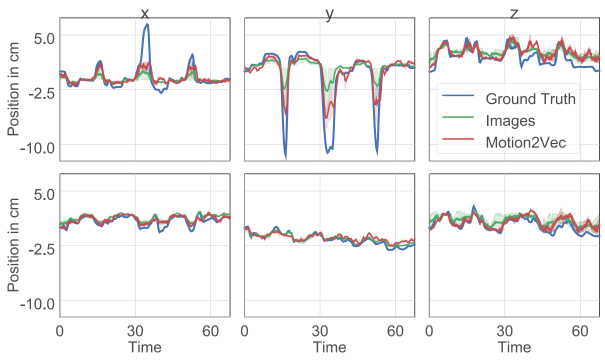


Fig. 13. Pose imitation on the da Vinci robot arms from the Motion2Vec embedded video sequences using a feedforward neural network for (top)left arm and (bottom) right arm in comparison with ground-truth and image-decoded poses. Results suggest 0.94 cm error in position per observation on the evaluation set. Ground-truth in blue, raw videos in green, predicted in red.

and following them with a LQT controller, we are able to autonomously perform manipulation tasks in a smooth manner. This has enabled a Baxter robot to tackle a pick-and-place via obstacle avoidance problem from previously unseen configurations of the environment. We extended the representation learning approach to videos with Motion2Vec that groups similar action segments in a deep embedding feature space, improving upon the interpretability and the segmentation performance over several state-of-the-art methods. We demonstrate its use on the dual-arm da Vinci robot arm to imitate surgical suturing poses from video demonstrations. In future work, we plan to learn closed-loop policies on the real robot from the embedded video representations. We are also interested in providing useful feedback for training and assistance in remote surgical procedures.

Acknowledgments

The first part on learning from trajectories was in part carried out at the Idiap Research Institute and Ecole Polytechnique Federale de Lausanne (EPFL), Switzerland, whereas the second part on learning from videos was in part carried out at the UC Berkeley in affiliation with the Berkeley AI Research (BAIR) Lab. The authors would like to thank Pierre Sermanet, Mariano Phielipp, Hanlin Tang, Raghav Anand, Brijen Thananjeyan, Sanjay Krishnan, Michael Laskey, and Roy Fox for their comments and feedback.

Funding

The author(s) disclosed receipt of the following financial support for the research, authorship, and/or publication of this article: This work was in part supported by the DexROV project through the EC Horizon 2020 program (grant number 635491), and the NSF (National Robotics Initiative Award 1734633 on Scalable Collaborative Human–Robot Learning (SCHooL)), along with donations from Intel, and by equipment grants from NVidia and

Intuitive Surgical. The information, data, comments, and views detailed herein may not necessarily reflect the endorsements of the sponsors.

Note

1. Setting $d_i = 0$ by choosing $\lambda_1 \gg 0$ gives the loss function formulation with isotropic Gaussian under SVA Roychowdhury et al., 2013).

ORCID iDs

Ajay Kumar Tanwani https://orcid.org/0000-0002-6365-8315
Jonathan Nicholas Lee https://orcid.org/0000-0002-1828-1707
Sylvain Calinon https://orcid.org/0000-0002-9036-6799

References

Ahmidi N, Tao L, Sefati S, et al. (2017) A dataset and benchmarks for segmentation and recognition of gestures in robotic surgery. *IEEE Transactions on Biomedical Engineering* 64(9): 2025–2041.

Argall BD, Chernova S, Veloso M and Browning B (2009) A survey of robot learning from demonstration. *Robotics and Autonomous Systems* 57(5): 469–483.

Asfour T, Azad P, Gyarfas F and Dillmann R (2008) Imitation learning of dual-arm manipulation tasks in humanoid robots. *International Journal of Humanoid Robotics* 5(2): 183–202.

Billard AG, Calinon S and Dillmann R (2016) Learning from humans. In: Siciliano B and Khatib O (eds.) *Handbook of Robotics*, 2nd ed. Secaucus, NJ: Springer, pp. 1995–2014.

Borrelli F, Bemporad A and Morari M (2011) *Predictive Control for Linear and Hybrid Systems*. Cambridge: Cambridge University Press.

Bowen C and Alterovitz R (2020) Probability-weighted temporal registration for improving robot motion planning and control learned from demonstrations. In: Morales M, Tapia L, Sánchez-Ante G and Hutchinson S (eds.) *Algorithmic Foundations of Robotics XIII*. Cham: Springer, pp. 246–263.

Broderick T, Kulis B and Jordan M (2013) MAD-Bayes: Map-based asymptotic derivations from Bayes. In: *International Conference on Machine Learning (ICML)*, pp. 226–234.

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

Calinon S and Lee D (2018) Learning control. In: Vadakkepat P and Goswami A (eds.) *Humanoid Robotics: A Reference*. Berlin: Springer.

DiPietro RS, Lea C, Malpani A, et al. (2016) Recognizing surgical activities with recurrent neural networks. *CoRR* abs/1606.06329.

Doersch C, Gupta A and Efros AA (2015) Unsupervised visual representation learning by context prediction. *CoRR* abs/1505.05192.

Duan Y, Andrychowicz M, Stadie B, et al. (2017) One-shot imitation learning. *CoRR* abs/1703.07326.

Duque DA, Prieto FA and Hoyos JG (2019) Trajectory generation for robotic assembly operations using learning by demonstration. *Robotics and Computer-Integrated Manufacturing* 57: 292–302.

Dwibedi D, Aytar Y, Tompson J, Sermanet P and Zisserman A (2019) Temporal cycle-consistency learning. *CoRR* abs/1904.07846.

Dwibedi D, Tompson J, Lynch C and Sermanet P (2018) Learning actionable representations from visual observations. *CoRR* abs/1808.00928.

Ebert F, Finn C, Dasari S, Xie A, Lee AX and Levine S (2018) Visual foresight: Model-based deep reinforcement learning for vision-based robotic control. *CoRR* abs/1812.00568.

Figueroa N and Billard A (2017) Transform-invariant nonparametric clustering of covariance matrices and its application to unsupervised joint segmentation and action discovery. *CoRR* abs/1710.10060.

Finn C and Levine S (2016) Deep visual foresight for planning robot motion. *CoRR* abs/1610.00696.

Finn C, Tan XY, Duan Y, Darrell T, Levine S and Abbeel P (2015) Learning visual feature spaces for robotic manipulation with deep spatial autoencoders. *CoRR* abs/1509.06113.

Finn C, Yu T, Zhang T, Abbeel P and Levine S (2017) One-shot visual imitation learning via meta-learning. In: *1st Conference on Robot Learning (CoRL 2017)*, Mountain View, CA.

Florence PR, Manuelli L and Tedrake R (2018) Dense object nets: Learning dense visual object descriptors by and for robotic manipulation. *CoRR* abs/1806.08756.

Fox R, Krishnan S, Stoica I and Goldberg K (2017) Multi-level discovery of deep options. *CoRR* abs/1703.08294.

Fox R, Shin R, Krishnan S, Goldberg K, Song D and Stoica I (2018) Parametrized hierarchical procedures for neural programming. In: *The International Conference on Learning Representations (ICLR'18)*.

Gales MJF (1999) Semi-tied covariance matrices for hidden Markov models. *IEEE Transactions on Speech and Audio Processing* 7(3): 272–281.

Gao Y, Vedula SS, Reiley CE, et al. (2014) Jhu-Isi gesture and skill assessment working set (JIGSAWS): A surgical activity dataset for human motion modeling. In: *MICCAI Workshop*.

Graves A (2012) *Supervised Sequence Labelling with Recurrent Neural Networks (Studies in Computational Intelligence)*. Berlin: Springer.

Hermans A, Beyer L and Leibe B (2017) In defense of the triplet loss for person re-identification. *CoRR* abs/1703.07737.

Ho J and Ermon S (2016) Generative adversarial imitation learning. *CoRR* abs/1606.03476.

Hochreiter S and Schmidhuber J (1997) Long short-term memory. *Neural Computation* 9(8): 1735–1780.

Hoque R, Seita D, Balakrishna A, et al. (2020) Visuospatial foresight for multi-step, multi-task fabric manipulation. *arXiv* 2003.09044.

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

Kappen HJ, Gómez V and Opper M (2012) Optimal control as a graphical model inference problem. *Machine Learning* 87(2): 159–182.

Koch G, Zemel R and Salakhutdinov R (2015) Siamese neural networks for one-shot image recognition. In: *Proceedings of the 32nd International Conference on Machine Learning*, Lille, France.

Krishnan S, Fox R, Stoica I and Goldberg K (2017) DDCO: Discovery of Deep Continuous Options forRobot Learning from Demonstrations. *CoRR* abs/1710.05421.

Krishnan S, Garg A, Patil S, et al. (2018) *Transition State Clustering: Unsupervised Surgical Trajectory Segmentation for Robot Learning*. Cham: Springer, pp. 91–110.

Kuehne H, Richard A and Gall J (2019) A hybrid RNN-HMM approach for weakly supervised temporal action segmentation. *CoRR* abs/1906.01028.

Kuehne H and Serre T (2015) Towards a generative approach to activity recognition and segmentation. *CoRR* abs/1509.01947.

Kulic D, Takano W and Nakamura Y (2008) Incremental learning, clustering and hierarchy formation of whole body motion patterns using adaptive hidden Markov chains. *The International Journal of Robotics Research* 27(7): 761–784.

Kulis B and Jordan M (2012) Revisiting k-means: New algorithms via Bayesian nonparametrics. In: *International Conference on Machine Learning ICML*, pp. 513–520.

Lafferty JD, McCallum A and Pereira FCN (2001) Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In: *Proceedings of the Eighteenth International Conference on Machine Learning (ICML '01)*. San Mateo, CA: Morgan Kaufmann, pp. 282–289.

Lea C, Hager GD and Vidal R (2015) An improved model for segmentation and recognition of fine-grained activities with application to surgical training tasks. In: *IEEE Winter Conference on Applications of Computer Vision*, pp. 1123–1129.

Lea C, Vidal R, Reiter A and Hager GD (2016) Temporal convolutional networks: A unified approach to action segmentation. *CoRR* abs/1608.08242.

Lee D and Ott C (2010) Incremental motion primitive learning by physical coaching using impedance control. In: *Proceedings IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Taipei, Taiwan, pp. 4133–4140.

Levine S (2018) Reinforcement learning and control as probabilistic inference: Tutorial and review. *CoRR* abs/1805.00909.

Liu Y, Gupta A, Abbeel P and Levine S (2017) Imitation from observation: Learning to imitate behaviors from raw video via context translation. *CoRR* abs/1707.03374.

McLachlan GJ, Peel D and Bean RW (2003) Modelling high-dimensional data by mixtures of factor analyzers. *Computational Statistics and Data Analysis* 41(3–4): 379–388.

Medina R J and Billard A (2017) Learning stable task sequences from demonstration with linear parameter varying systems and hidden Markov models. In: *Conference on Robot Learning (CoRL)*.

Misra I, Zitnick CL and Hebert M (2016) Unsupervised learning using sequential verification for action recognition. *CoRR* abs/1603.08561.

Mor B, Garhwal S and Kumar A (2021) A systematic review of hidden Markov models and their applications. *Archives of Computational Methods in Engineering* 28: 1429–1448.

Murali A, Garg A, Krishnan S, et al. (2016) TSC-DL: Unsupervised trajectory segmentation of multi-modal surgical demonstrations with deep learning. In: *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 4150–4157.

Nehaniv CL and Dautenhahn K (eds.) (2004) *Imitation and Social Learning in Robots, Humans, and Animals: Behavioural, Social and Communicative Dimensions*. Cambridge: Cambridge University Press.

Niekum S, Osentoski S, Konidaris G and Barto AG (2012) Learning and generalization of complex tasks from unstructured demonstrations. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5239–5246.

Osa T, Pajarinen J, Neumann G, Bagnell A, Abbeel P and Peters J (2018) *An Algorithmic Perspective on Imitation Learning*. Now Publishers Inc.

Padoy N and Hager GD (2011) Human-machine collaborative surgery using learned models. In: *IEEE International Conference on Robotics and Automation*, pp. 5285–5292.

Paraschos A, Daniel C, Peters JR and Neumann G (2013) Probabilistic movement primitives. *Advances in Neural Information Processing Systems* 26: 2616–2624.

Pervez A and Lee D (2018) Learning task-parameterized dynamic movement primitives using mixture of GMMs. *Intelligent Service Robots* 11(1): 61–78.

Pignat E and Calinon S (2017) Learning adaptive dressing assistance from human demonstration. *Robotics and Autonomous Systems* 93: 61–75.

Rabiner LR (1989) A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE* 77(2): 257–285.

Rippel O, Paluri M, Dollar P and Bourdev L (2015) Metric learning with adaptive density discrimination. *arXiv preprint arXiv:1511.05939*.

Ross S, Gordon GJ and Bagnell JA (2011) A reduction of imitation learning and structured prediction to no-regret online learning. In: *International Conference on Artificial Intelligence and Statistics*.

Roychowdhury A, Jiang K and Kulis B (2013) Small-variance asymptotics for hidden Markov models. *Advances in Neural Information Processing Systems* 26: 2103–2111.

Rozo L, Guo M, Kupcsik AG, et al. (2020) Learning and sequencing of object-centric manipulation skills for industrial tasks. *arXiv* 2008.10471.

Schaal S, Ijspeert A and Billard A (2003) Computational approaches to motor learning by imitation. *Philosophical Transactions of the Royal Society of London: Series B, Biological Sciences* 358(1431): 537–547.

Schmidt T, Newcombe R and Fox D (2017) Self-supervised visual descriptor learning for dense correspondence. *IEEE Robotics and Automation Letters* 2(2): 420–427.

Schroff F, Kalenichenko D and Philbin J (2015) Facenet: A unified embedding for face recognition and clustering. *CoRR* abs/1503.03832.

Schulman J, Gupta A, Venkatesan S, Tayson-Frederick M and Abbeel P (2013) A case study of trajectory transfer through non-rigid registration for a simplified suturing scenario. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 4111–4117.

Sen S, Garg A, Gealy DV, McKinley S, Jen Y and Goldberg K (2016) Automating multi-throw multilateral surgical suturing with a mechanical needle guide and sequential convex optimization. In: *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 4178–4185.

Sermanet P, Lynch C, Hsu J and Levine S (2017) Time-contrastive networks: Self-supervised learning from multi-view observation. *CoRR* abs/1704.06888.

Shiarlis K, Wulfmeier M, Salter S, Whiteson S and Posner I (2018) TACO: Learning task decomposition via temporal alignment for control. In: *International Conference on Machine Learning (ICML'18)*.

Sohn K (2016) Improved deep metric learning with multi-class n-pair loss objective. *Advances in Neural Information Processing Systems* 29: 1857–1865.

Sundaresan P, Grannen J, Thananjeyan B, et al. (2020) Learning rope manipulation policies using dense object descriptors trained on synthetic depth data. in: *2020 International Conference on Robotics and Automation*.

Sutton C and McCallum A (2012) An introduction to conditional random fields. *Foundation and Trends in Machine Learning* 4(4): 267–373.

Tang K (2012) Learning latent temporal structure for complex event detection. In: *Proceedings of the 2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR '12)*, pp. 1250–1257.

Tanwani AK (2018) *Generative Models for Learning Robot Manipulation Skills from Humans*. PhD Thesis, Ecole Polytechnique Federale de Lausanne, Switzerland.

Tanwani AK and Calinon S (2016) Learning robot manipulation tasks with task-parameterized semitied hidden semi-Markov model. *IEEE Robotics and Automation Letters* 1(1): 235–242.

Tanwani AK and Calinon S (2017) A generative model for intention recognition and manipulation assistance in teleoperation. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 43–50.

Tanwani AK and Calinon S (2019) Small-variance asymptotics for non-parametric online robot learning. *The International Journal of Robotics Research* 38(1): 3–22.

Tanwani AK, Lee J, Thananjeyan B, et al. (2018) Generalizing robot imitation learning with invariant hidden semi-Markov models. In: *Algorithmic Foundations of Robotics XIII (WAFR 2018)* (*Springer Proceedings in Advanced Robotics*, Vol. 14). Cham: Springer, pp. 196–211.

Tanwani AK, Sermanet P, Yan A, Anand R, Phielipp M and Goldberg K (2020) Motion2vec semi-supervised representation learning from surgical videos. In: *Proceedings IEEE Internationall Conference on Robotics and Automation (ICRA)*, pp. 1–8.

Teh YW, Jordan MI, Beal MJ and Blei DM (2006) Hierarchical Dirichlet processes. *Journal of the American Statistical Association* 101(476): 1566–1581.

Ti B, Gao Y, Li Q and Zhao J (2019) Human intention understanding from multiple demonstrations and behavior generalization in dynamic movement primitives framework. *IEEE Access* 7: 36186–36194.

Tipping ME and Bishop CM (1999) Mixtures of probabilistic principal component analyzers. *Neural Computation* 11(2): 443–482.

Todorov E (2008) General duality between optimal control and estimation. In: *47th IEEE Conference on Decision and Control*, pp. 4286–4292.

Torabi F, Warnell G and Stone P (2018) Behavioral cloning from observation. *CoRR* abs/1805.01954.

Toussaint M (2009) Robot trajectory optimization using approximate inference. In: *Proceedings of the 26th Annual International Conference on Machine Learning (ICML '09)*. New York: ACM Press, pp. 1049–1056.

Vakanski A, Mantegh I, Irish A and Janabi-Sharifi F (2012) Trajectory learning for robot programming by demonstration using hidden Markov model and dynamic time warping. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* 42(4): 1039–1052.

Wang X and Gupta A (2015) Unsupervised learning of visual representations using videos. *CoRR* abs/1505.00687.

Wang Y and Zhu J (2015) DP-space: Bayesian nonparametric subspace clustering with small-variance asymptotics. In: *Proceedings of the 32nd International Conference on Machine Learning (ICML)*, pp. 862–870.

Wilson AD and Bobick AF (1999) Parametric hidden Markov models for gesture recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 21(9): 884–900.

Wolpert DM, Diedrichsen J and Flanagan JR (2011) Principles of sensorimotor learning. *Nature Reviews* 12: 739–751.

Xu D, Nair S, Zhu Y, et al. (2017) Neural task programming: Learning to generalize across hierarchical tasks. *CoRR* abs/1710.01813.

Yang C, Luo J, Liu C, Li M and Dai S (2019) Haptics electromyography perception and learning enhanced intelligence for teleoperated robot. *IEEE Transactions on Automation Science and Engineering* 16(4): 1512–1521.

Young S, Gandhi D, Tulsiani S, Gupta A, Abbeel P and Pinto L (2020) Visual imitation made easy. *arXiv* 2008.04899.

Yu SZ (2010) Hidden semi-Markov models. *Artificial Intelligence* 174: 215–243.

Zeng A, Song S, Nießner M, Fisher M, Xiao J and Funkhouser T (2017) 3dmatch: Learning local geometric descriptors from RGB-d reconstructions. In: *CVPR*.

Zhang Y, Pezeshki M, Brakel P, et al. (2017) Towards end-to-end speech recognition with deep convolutional neural networks. *CoRR* abs/1701.02720.

Zhao H, Yuen PC and Kwok JT (2006) A novel incremental principal component analysis and its application for face recognition. *Transactions on Systems, Man, and Cybernetics Part B* 36(4): 873–886.

Appendix A. Linear quadratic tracking

The discrete-time dynamical system for the double integrator is defined as

$$\overbrace{\begin{bmatrix} \boldsymbol{x}_{t+1} \\ \boldsymbol{x}_{t+2} \end{bmatrix}}^{\boldsymbol{\xi}_{t+1}} = \overbrace{\begin{bmatrix} \boldsymbol{I} & \boldsymbol{\Delta} t \\ \boldsymbol{0} & \boldsymbol{I} \end{bmatrix}}^{\boldsymbol{A}_d} \overbrace{\begin{bmatrix} \boldsymbol{x}_t \\ \boldsymbol{x}_{t+1} \end{bmatrix}}^{\boldsymbol{\xi}_t} + \overbrace{\begin{bmatrix} \boldsymbol{I}\frac{1}{2}\Delta t^2 \\ \boldsymbol{I}\Delta t \end{bmatrix}}^{\boldsymbol{B}_d} \boldsymbol{u}_t \tag{23}$$

The control law $\boldsymbol{u}_t^*$ that minimizes the cost function in (10) under *finite horizon* subject to the linear dynamics in discrete time is given as

$$\begin{aligned} \boldsymbol{u}_t^* = &- \left(\boldsymbol{R} + \boldsymbol{B}_d^{\mathrm{T}} \boldsymbol{P}_t \boldsymbol{B}_d\right)^{-1} \boldsymbol{B}_d^{\mathrm{T}} \boldsymbol{P}_t \boldsymbol{A}_d (\boldsymbol{\xi}_t - \hat{\boldsymbol{\mu}}_t) \\ &- \left(\boldsymbol{R} + \boldsymbol{B}_d^{\mathrm{T}} \boldsymbol{P}_t \boldsymbol{B}_d\right)^{-1} \boldsymbol{B}_d^{\mathrm{T}} (\boldsymbol{P}_t (\boldsymbol{A}_d \hat{\boldsymbol{\mu}}_t - \hat{\boldsymbol{\mu}}_t) + \boldsymbol{d}_t) \end{aligned} \tag{24}$$

$$\begin{aligned} \boldsymbol{u}_t^* = &\boldsymbol{K}_t^{\mathcal{P}} (\hat{\boldsymbol{\mu}}_t^x - \boldsymbol{x}_t) + \boldsymbol{K}_t^{\mathcal{V}} (\hat{\boldsymbol{\mu}}_t^{\dot{x}} - \dot{\boldsymbol{x}}_t) \\ &- \left(\boldsymbol{R} + \boldsymbol{B}_d^{\mathrm{T}} \boldsymbol{P}_t \boldsymbol{B}_d\right)^{-1} \boldsymbol{B}_d^{\mathrm{T}} (\boldsymbol{P}_t (\boldsymbol{A}_d \hat{\boldsymbol{\mu}}_t - \hat{\boldsymbol{\mu}}_t) + \boldsymbol{d}_t) \end{aligned} \tag{25}$$

where $[\boldsymbol{K}_t^{\mathcal{P}}, \boldsymbol{K}_t^{\mathcal{V}}] = -\left(\boldsymbol{R} + \boldsymbol{B}_d^{\mathrm{T}} \boldsymbol{P}_t \boldsymbol{B}_d\right)^{-1} \boldsymbol{B}_d^{\mathrm{T}} \boldsymbol{P}_t \boldsymbol{A}_d$ are the full stiffness and damping matrices for the feedback term, and $\left(\boldsymbol{R} + \boldsymbol{B}_d^{\mathrm{T}} \boldsymbol{P}_t \boldsymbol{B}_d\right)^{-1} \boldsymbol{B}_d^{\mathrm{T}} (\boldsymbol{P}_t (\boldsymbol{A}_d \hat{\boldsymbol{\mu}}_t - \hat{\boldsymbol{\mu}}_t) + \boldsymbol{d}_t)$ is the feedforward term. Here $\boldsymbol{P}_t$ and $\boldsymbol{d}_t$ are obtained by solving the Riccati differential equation and linear differential equation backwards in discrete time from terminal conditions $\boldsymbol{P}_{T_p} = \boldsymbol{Q}_{T_p}$ and $\boldsymbol{d}_{T_p} = \boldsymbol{0}$, respectively,

$$\begin{aligned}\boldsymbol{P}_{t-1} &= \boldsymbol{Q}_t - \boldsymbol{A}_d^{\mathrm{T}}(\boldsymbol{P}_t\boldsymbol{B}_d(\boldsymbol{R}+\boldsymbol{B}_d^{\mathrm{T}}\boldsymbol{P}_t\boldsymbol{B}_d)^{-1}\boldsymbol{B}_d^{\mathrm{T}}\boldsymbol{P}_t - \boldsymbol{P}_t)\boldsymbol{A}_d \\ \boldsymbol{d}_{t-1} &= (\boldsymbol{A}_d^{\mathrm{T}} - \boldsymbol{A}_d^{\mathrm{T}}\boldsymbol{P}_t\boldsymbol{B}_d(\boldsymbol{R}+\boldsymbol{B}_d^{\mathrm{T}}\boldsymbol{P}_t\boldsymbol{B}_d)^{-1}\boldsymbol{B}_d^{\mathrm{T}}) \\ &\quad (\boldsymbol{P}_t(\boldsymbol{A}_d\hat{\boldsymbol{\mu}}_t - \hat{\boldsymbol{\mu}}_{t+1}) + \boldsymbol{d}_t)\end{aligned} \tag{26}$$

For the *infinite horizon* case with $T \to \infty$ and the desired pose $\hat{\boldsymbol{\mu}}_t = \hat{\boldsymbol{\mu}}_{t_0}$, the control law in (24) remains the same except the feedforward term is set to zero and $\boldsymbol{P}_{t-1} = \boldsymbol{P}_t = \boldsymbol{P}$ is the steady-state solution obtained by eigenvalue decomposition of the discrete algebraic Riccati equation (DARE) in (26) (Borrelli et al., 2011). To solve DARE, we define the symplectic matrix,

$$\boldsymbol{H}_b = \begin{bmatrix} \boldsymbol{A}_d + \boldsymbol{B}_d\boldsymbol{R}^{-1}\boldsymbol{B}_d^{\mathrm{T}}(\boldsymbol{A}_d^{-1})^{\mathrm{T}}\boldsymbol{Q} & \boldsymbol{B}_d\boldsymbol{R}^{-1}\boldsymbol{B}_d^{\mathrm{T}}(\boldsymbol{A}_d^{-1})^{\mathrm{T}} \\ -(\boldsymbol{A}_d^{-1})^{\mathrm{T}}\boldsymbol{Q} & (\boldsymbol{A}_d^{-1})^{\mathrm{T}} \end{bmatrix} \tag{27}$$

The eigenvectors of $\boldsymbol{H}_b$ corresponding to eigenvalues lying inside the unit circle are used to solve DARE. Let $\begin{bmatrix} \boldsymbol{V}_1^{\mathrm{T}} & \boldsymbol{V}_{21}^{\mathrm{T}} \end{bmatrix}^{\mathrm{T}}$ denote the corresponding subspace of $\boldsymbol{H}_b$, then the solution of DARE is $\boldsymbol{P} = \boldsymbol{V}_{21}\boldsymbol{V}_1^{-1}$ and the control law takes the form

$$\boldsymbol{u}_t^* = -(\boldsymbol{R}+\boldsymbol{B}_d^{\mathrm{T}}\boldsymbol{P}\boldsymbol{B}_d)^{-1}\boldsymbol{B}_d^{\mathrm{T}}\boldsymbol{P}\boldsymbol{A}_d(\boldsymbol{\xi}_t - \hat{\boldsymbol{\mu}}_t) \tag{28}$$

Both discrete and continuous time linear quadratic regulators/trackers can be used to follow the desired pose/trajectory. The discrete time formulation, however, gives numerically stable results for a wide range of values of $\boldsymbol{R}$.

Appendix B. Distance to cluster subspace versus distance to cluster mean

The distance of a datapoint $\boldsymbol{\xi}_t$ to an existing cluster with mean $\boldsymbol{\mu}_i$ is represented as $\| \boldsymbol{\xi}_t - \boldsymbol{\mu}_i \|_2^2$. In contrast, we define the distance of a datapoint from the subspace of a cluster, $\mathrm{dist}(\boldsymbol{\xi}_t, \boldsymbol{\mu}_i, \boldsymbol{U}_i^{d_i})^2$, as the difference between the mean-centered datapoint and the mean-centered datapoint projected upon the subspace $\boldsymbol{U}_i^{d_i} \in \mathbb{R}^{D \times d_i}$ spanned by the d_i unit eigenvectors of the covariance matrix, i.e.,

$$\mathrm{dist}(\boldsymbol{\xi}_t, \boldsymbol{\mu}_i, \boldsymbol{U}_i^{d_i}) = \| (\boldsymbol{\xi}_t - \boldsymbol{\mu}_i) - \rho_i \boldsymbol{U}_i^{d_i}\boldsymbol{U}_i^{d_i\mathrm{T}}(\boldsymbol{\xi}_t - \boldsymbol{\mu}_i)\|_2 \tag{29}$$

where

$$\rho_i = \exp\left(-\frac{\| \boldsymbol{\xi}_t - \boldsymbol{\mu}_i \|_2^2}{b_m}\right)$$

weighs the projected mean-centered datapoint according to the distance of the datapoint from the cluster center ($0 < \rho_i \leq 1$). Its effect is controlled by the bandwidth parameter b_m. If b_m is large, then the far away clusters have a greater influence; otherwise, nearby clusters are favored. Note that ρ_i assigns more weight to the projected mean-centered datapoint for the nearby clusters than the distant clusters to limit the size of the cluster/subspace. Our subspace distance formulation is different from Wang and Zhu (2015) as we weigh the subspace of the nearby clusters more than the distant clusters. This allows us to avoid clustering all the datapoints in the same subspace (near or far) together.