Article

The International Journal of Robotics Research
2017, Vol. 36(13–14) 1595–1618
© The Author(s) 2017
Reprints and permissions:
sagepub.co.uk/journalsPermissions.nav
DOI: 10.1177/0278364917743319
journals.sagepub.com/home/ijr
SAGE

Transition state clustering: Unsupervised surgical trajectory segmentation for robot learning

Sanjay Krishnan[1,*], Animesh Garg[1,*], Sachin Patil[1], Colin Lea[2], Gregory Hager[2], Pieter Abbeel[1] and Ken Goldberg[1]

Abstract
Demonstration trajectories collected from a supervisor in teleoperation are widely used for robot learning, and temporally segmenting the trajectories into shorter, less-variable segments can improve the efficiency and reliability of learning algorithms. Trajectory segmentation algorithms can be sensitive to noise, spurious motions, and temporal variation. We present a new unsupervised segmentation algorithm, transition state clustering (TSC), which leverages repeated demonstrations of a task by clustering segment endpoints across demonstrations. TSC complements any motion-based segmentation algorithm by identifying candidate transitions, clustering them by kinematic similarity, and then correlating the kinematic clusters with available sensory and temporal features. TSC uses a hierarchical Dirichlet process Gaussian mixture model to avoid selecting the number of segments a priori. *We present simulated results to suggest that TSC significantly reduces the number of false-positive segments in dynamical systems observed with noise as compared with seven probabilistic and non-probabilistic segmentation algorithms. We additionally compare algorithms that use piecewise linear segment models, and find that TSC recovers segments of a generated piecewise linear trajectory with greater accuracy in the presence of process and observation noise. At the maximum noise level, TSC recovers the ground truth 49% more accurately than alternatives. Furthermore, TSC runs 100× faster than the next most accurate alternative autoregressive models, which require expensive Markov chain Monte Carlo (MCMC)-based inference. We also evaluated TSC on 67 recordings of surgical needle passing and suturing. We supplemented the kinematic recordings with manually annotated visual features that denote grasp and penetration conditions. On this dataset, TSC finds 83% of needle passing transitions and 73% of the suturing transitions annotated by human experts.*

Keywords
trajectory segmentation, surgical robotics, robot learning

1. Introduction

Describing a complex task in terms of motion primitives has been an important area of research since the early days of robotic planning (Brooks, 1986; Fikes et al., 1972). An important sub-problem of primitive-based task planning is *segmentation*, where given a set of observation trajectories, one needs to identify the start and end times of the underlying primitives in each trajectory (for an overview, see Lin et al., 2016). The segmentation problem is crucial for analyzing and modeling expert demonstration data (Argall et al., 2009), since it facilitates learning localized control policies (Konidaris et al., 2011; Murali et al., 2015; Niekum et al., 2012), and adaptation to unseen scenarios (Ijspeert et al., 2002; Ude et al., 2010).

While one could infer segmentation criteria from manual annotations or matching to pre-defined dictionaries of motion templates, labeling consistency and supervisory burden are concerns in *supervised* approaches. Complex, high-dimensional data can require a large amount of labels before a viable segmentation model can be learned. Similarly, dictionaries of primitives can be incomplete. To avoid these problems, *unsupervised* segmentation methods have long been studied (Morasso, 1983; Sternad and Schaal, 1999; Viviani and Cenzato, 1985). Recently, several new probabilistic approaches have been proposed that pose segmentation as a probabilistic inference problem (Alvarez et al., 2010; Barbič et al., 2004; Chiappa and Peters, 2010; Krüger et al., 2012; Niekum et al., 2012; Wächter and

[1]University of California, Berkeley, USA
[2]Johns Hopkins University, USA
[*]These authors contributed equally.

Corresponding author:
Sanjay Krishnan, Soda Hall, University of California–Berkeley, CA 94720, USA.
Email: sanjay@eecs.berkeley.edu

Asfour, 2015). The approaches model a trajectory as generated from a mixture of parametrized dynamical regimes, and an inference procedure learns the dynamical parameters that can be used to identify time segments at which each regime is active.

Explicitly modeling the dynamics can require a large number of parameters to be learned. This makes such approaches somewhat sensitive to any noise in the dataset, especially when the datasets are small. This sensitivity leads to challenges in applications such as robotic surgery. The adoption of robot-assisted minimally invasive surgery (RMIS) generating datasets of kinematic and video recordings of surgical procedures (Gao et al., 2014), and where trajectories are collected from teleoperation interfaces. With such interfaces, we have observed significant jitter in motion and noise due to time delays Figure 1 plots 10 expert demonstrations of a surgical training task (Chuck et al., 2017; Liang et al., 2017). In such a setting, the robustness and stability of the segmentation algorithm are a key concern in surgical segmentation, and to the best of the authors' knowledge, prior work mitigates this issue by leveraging pre-defined dictionaries of motion segments (Lin et al., 2005, 2006).

In many important tasks, while the demonstration motions may vary and be noisy, each demonstration contains roughly the same order of true segments, e.g. well-defined surgical training procedures. This consistent, repeated structure can be exploited to infer global segmentation criteria. By assuming known sequential segment-to-segment transitions, the problem reduces to identifying a common set of segment-to-segment transition events, not corresponding to the entirety of trajectory segments across the whole dataset. This allows us to apply coarser, imperfect motion-based segmentation algorithms first that create a large set of candidate transitions. Then, we can filter this set by identifying transition events that occurred at similar times and states. Our experiments suggest that this approach has improved robustness and sample efficiency, while approximating the behavior of more complicated dynamical systems-based approaches in many real problems.

This paper formalizes this intuition into a new hierarchical clustering algorithm for unsupervised segmentation called transition state clustering (TSC). The proposed approach is also relevant to problems in other domains, but this paper will focus on results from surgical applications. TSC first applies a motion-based segmentation model over the noisy trajectories and identifies a set of candidate segment transitions in each. TSC then clusters the transition states (states at times transitions occur) in terms of kinematic, sensory, and temporal similarity. The clustering process is hierarchical where the transition states are first assigned to Gaussian mixture clusters according to kinematic state, then these clusters are sub-divided using the sensory features, and finally by time. We present experiments where these sensory features are constructed from video. The learned clustering model can be applied to segment previously unseen trajectories by the same global criteria. To avoid setting the number of clusters at each level of the hierarchy in advance, the number of regions are determined by a Dirichlet process (DP) prior. A series of merging and pruning steps remove sparse clusters and repetitive loops.

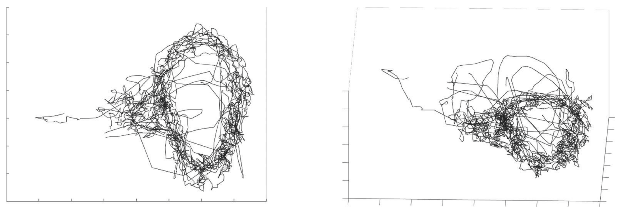


Fig. 1. Plot of 10 trajectories of the end-effector (x, y, z) positions on an identical circle cutting task on the da Vinci Research Kit (dVRK). This plot illustrates the variability of demonstrations even when the task is identical.

Example: As an example of how noise can affect segmentation, consider a system where a spherical ball is dropped until it bounces off a block. Under noiseless conditions, most classical segmentation techniques that look for changes in direction (e.g. zero-velocity crossings) or local linearity of motion would identify two segments (Figure 2). If the observations are perturbed by noise, these approaches tend to "over segment," where noise can be confused for actual changes in direction. If we collect five demonstrations from the same system, and plot the estimated segment transitions (state of the end point) for each of the noisy demonstrations: we would find that the densest clusters correspond to actual segment endpoints. TSC exploits this property to improve the robustness of motion-based segmentation.

This paper is a substantially revised and expanded version of Krishnan et al. (2015). This version includes several new experiments evaluating TSC against a broader set of alternative algorithms, an expanded technical discussion about the model, and revised accuracy metrics consistent with recent segmentation work in robotics and computer vision.

2. Related work

This section provides an overview of the development of unsupervised segmentation with a focus on the recent trend of probabilistic models. The study of trajectory segmentation algorithms has a long history, especially in the context of decomposing human motion into primitives. For example, Viviani and Cenzato (1985) explored using the "two-thirds" powerlaw coefficient to determine segment boundaries in handwriting. Similarly, Morasso (1983) showed that rhythmic 3D motions of a human arm could be modeled as piecewise linear. Mao et al. (2014) considered a segmentation model that classified human arm motions into three phases: (1) a reach phase where the arm moves until it comes in contact with an object; (2) a manipulation phase

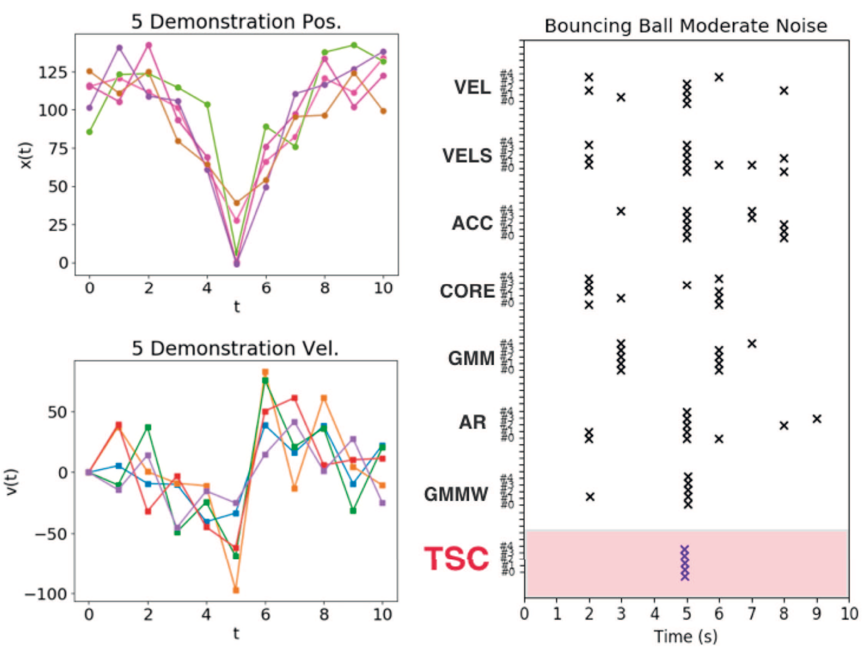


Fig. 2. Challenges of unsupervised segmentation in the presence of noise. A change in velocity condition can lead to spurious segments. Clustering segment end-points over repeated demonstrations can help remove artifacts of noise as in TSC.

where the arm manipulates the object; and (3) a withdraw phase where the arm releases the object. Other approaches rely on thresholds on metrics such as curvature changes, zero-velocity crossings, and jerk criteria (Faria et al., 2012; Fod et al., 2002; Sahbani et al., 2008). Temporal segmentation is also studied in the motion capture community (Moeslund and Granum, 2001). Motivated by segmenting human motions, these models make strong assumptions kinematics of end-effector that generates the motions. This motivates the development of probabilistic segmentation models that can estimate local structure to mitigate such assumptions.

2.1. *Fully unsupervised approaches*

One approach is to use a mixture model, and many unsupervised segmentation techniques are based on hidden Markov models (HMMs) with Gaussian emissions (Asfour et al., 2008; Calinon et al., 2010; Gehrig et al., 2011; Kruger et al., 2010; Kulic et al., 2009; Vakanski et al., 2012). Consider a continuous-time vector-valued trajectory, which is a sequence of T vectors x_t in some vector space $\mathbb{R}^p$. One can write a generative model where the observed trajectory is composed of a sequence of k latent primitive behaviors. These behaviors have Markovian dynamics and transition based on a latent Markov chain, and conditioned on the current behavior the observed state is drawn from a particular Gaussian mixture component. One challenge is that tuning the number of segments is a key hyper-parameter. This basic model was extended by Krüger et al. (2012) to flexibly set the number of mixture components using a DP. Lee et al. (2015) further explored using dimensionality reduction and tune the number of mixture components using the Bayesian information criterion.

This basic logic has been extended to more complex transition dynamics such as the hidden semi-Markov model, which additionally models the amount of time spent in a given state (Tanwani and Calinon, 2016). There are also extensions that consider changes to emission model in addition to the high-level transitions. Willsky et al. (2009) proposed a beta process autoregressive HMM, which has also been applied by in robotics (Niekum and Chitta, 2013; Niekum et al., 2012). This model fits an autoregressive model to time series, where $\mathbf{x}_{t+1}$ is a linear function of states $\mathbf{x}_{t-k}, \ldots, \mathbf{x}_t$. The linear function switches according to a HMM with states parametrized by a beta-Bernoulli model (i.e. beta process).

It is important to note that the HMM-based approaches were first designed in the context of signal processing. The goal was to segment a long continuous stream of observations into a relatively small number of distinct symbols. In these HMM approaches, one learns the high-level transition structure as well as the low-level primitives. Demonstration data differs from this context as it is episodic in nature (multiple repeated demonstrations), may have a large number of primitives, and a relatively fixed transition structure. As a result we find that TSC, which explicitly leverages this structure, is more data efficient and robust to un-modeled noise. This weakness of HMMs is well-established, where “the low-level dynamics within a segment are more structured and predictable than the higher-level dynamics that govern transitions between segments” (Saeedi et al., 2016).

We compare against several representative approaches from this family of algorithms and find that TSC is more robust for small noisy datasets.

There are also several control-theoretic formulations of segmentation (Alvarez et al., 2010, 2013; Sternad and Schaal, 1999). In a seminal paper, Sternad and Schaal (1999) provided a formal framework for control-theoretic segmentation of trajectories. Alvarez et al. (2010) used a latent-force model, where variables are modeled as coupled by a dynamical system. Switching behavior in the coupling system are used to derive segmentation criteria. TSC can be interpreted as fitting local linear dynamics models to the demonstrations, and to improve robustness, we have multiple levels of clustering and pruning steps that remove sparse clusters. However, in this paper, we primarily consider kinematics and hope to explore segmentation with dynamics in the future.

2.2. Label-based approaches

There are also several methods that leverage labeled data to segment trajectories. That is, an expert annotates a small set of trajectories with labels and a model extrapolates segmentation properties on unlabeled trajectories (Lea et al., 2015; Lin et al., 2014; Quellec et al., 2014; Tao et al., 2013; Varadarajan et al., 2009; Zappella et al., 2013). This approach is very common in robotic surgery; however, it involves the time-consuming the process of identifying surgemes in existing data sources for use as training and testing data (Lea et al., 2015; Tao et al., 2013; Varadarajan et al., 2009). For example, given manually segmented videos, Zappella et al. (2013) used features from both the videos and kinematic data to classify surgical motions. Similarly, Quellec et al. (2014) use manually segmented examples as training for segmentation and recognition of surgical tasks based on archived cataract surgery videos.

2.3. Dictionary or library-based approaches

An alternative viewpoint to the fully unsupervised approaches is to have a small library of primitive templates that are modulated and combined to construct more complex behaviors (Chiappa and Peters, 2010; Meier et al., 2011; Pastor et al., 2013). Chiappa and Peters (2010) generated the observations from a set of movement templates. The observed time series is generated from a composition of these templates modulated by a noisy transformation. Parameter inference can be done with a Markov chain Monte Carlo (MCMC) method or a variational approximation. Some of these approaches use a approach based on dynamic movement primitives (DMPs) (Schaal, 2006). DMPs are a mathematical formalization of the composition of primitives. Each DMP is a nonlinear dynamical system with well-specified, stable behavior and a forcing term that makes it follow a trajectory of interest. There are also a number of interesting challenges in applying such approaches when the trajectories are periodic or have a rotational topology (Ude et al., 2010), and when the primitives are probabilistic (Paraschos et al., 2013). Likewise, the surgical robotics literature has also often leveraged a predefined dictionary called surgemes (Lin et al., 2005, 2006). These surgemes were constructed through domain expertise and careful examination of expert surgeons. TSC can be seen as automatically bootstrapping the library of primitives through clustering. It first takes a coarse model that detects transitions and clusters this into corresponded segments.

2.4. Two-step segmentation

While library-based approaches have been very popular, building a library of primitives can be challenging. One way to address this problem is to have a two-step procedure. The first step applies a less-complex model that over-segments each trajectory, i.e. a set candidate segment endpoints that are a superset of the true segment endpoints. For example, Meier et al. (2012) found minima in the velocity and/or acceleration profile of the trajectories. These candidates describe potential segment endpoints, but they do not assign the segment labels to any partial trajectories. These approaches then apply a subsequent probabilistic inference step that refines the candidates (Lioutikov et al., 2015; Meier et al., 2012). Both of these works are very similar in spirit to TSC. In both, the first step generates candidates (that we call transition states), which are refined by a probabilistic model (clustering). We additionally present an option for automatically generating these initial candidates using GMMs. We find that this approach experimentally works across a number of domains. The particulars of the refinement step are different in all of the works. For example, we use a hierarchical DP mixture model, Meier et al. (2012) fitted a parametrized DMP model and filters for the most likely segment candidates, and Lioutikov et al. (2015) used a probabilistic movement primitives model with a message passing inference algorithm. In this work, we did not consider the relationship between TSC and DMPs, and we hope to compare to these two approaches in future work.

2.5. TSC contributions

Segmentation is a very well-studied field and TSC certainly builds on several seminal works, such as Calinon et al. (2010) and Krüger et al. (2012). The goal of this paper is to explore to what extent an unsupervised approach (no dictionary and no labels) can segment realistic surgical data reliably. To the best of the authors' knowledge, prior work in surgical robotics has only considered supervised segmentation using either segmented examples or dictionaries.

We present a framework that uses GMMs to coarsely segment a trajectory and hierarchical clustering to correspond segments across trajectories. TSC makes a key assumption about sequential segment-to-segment transitions, and the problem reduces to identifying a common set of segment-to-segment transition events, not corresponding to the entirety of the trajectory segments across the whole dataset. We present a through experimental comparison of deterministic models, probabilistic models, and replacing various components of TSC.

3. Problem statement and definitions

This section describes the problem setting, assumptions, and notation. Let $D = \{d_i\}$ be a set of demonstrations of a robotic task. Each demonstration of a task d is a discrete-time sequence of T state vectors in a feature-space $\mathcal{X}$. The feature space is a concatenation of kinematic features X (e.g. robot position) and sensory features V (e.g. visual features from the environment).

Definition 1 (Segmentation)**.** *A segmentation of a task is defined as a function* **S** *that assigns each state in every demonstration trajectory to an integer* $1, 2, \ldots, k$*:*

$$\mathcal{S} : d \mapsto (a_n)_{1,\ldots,|d|}\,, \quad a_n \in 1, \ldots, k$$

and $\mathcal{S}$ *is a non-decreasing function in time (no repetitions).*

3.1. Candidate transitions

Suppose we are given a function that just identifies candidate segment endpoints based on the kinematic features. Such a function is weaker than a segmentation function since it does not globally label the detected segments. This leads to the following definition.

Definition 2 (Transition indicator function)**.** *A transition indicator function* **T** *is a function that maps each kinematic state in a demonstration d to* $\{0, 1\}$*:*

$$\mathbf{T} : d \mapsto (a_n)_{1,\ldots,|d|}\,, \quad a_n \in 0, 1$$

The above definition naturally leads to a notion of transition states, the states and times at which transitions occur.

Definition 3 (Transition states)**.** *For a demonstration* d_i*, let* $o_{i,t}$ *denote the kinematic state, visual state, and time* (x, v, t) *at time t. Transition states are the set of state–time tuples where the indicator is 1:*

$$\Gamma = \bigcup_{i}^{N} \{o_{i,t} \in d_i : \mathbf{T}(d_i)_t = 1\}$$

The goal of TSC is to take the transition states Γ and recover a segmentation function **S**. This segmentation function is stronger than the provided **T** since it not only indicates that a transition has occurred but labels the segment transition consistently across demonstrations.

3.2. Assumptions

We assume that all possible true segments are represented in each demonstration by at least one transition (some might be false positives). Given the segmentation function $\mathcal{S}(d_i)$, one can define a set of *true* transition states:

$$\Gamma^* = \{o_{i,t} \in d_i : \mathcal{S}(d_i)_{t-1} \neq \mathcal{S}(d_i)_t\,,\ t > 0\}$$

These satisfy the following property:

$$\Gamma^* \subseteq \Gamma$$

In other words, we assume that a subset of transition states discovered by the indicator function correspond with the true segment transitions. There can be false positives but no false negatives (a demonstration where a segment transition is missed by the transition indicator function). Since the segmentation function is sequential and in a fixed order, this leads to a model where we are trying to find the $k - 1$ true segment–segment transition points in Γ.

3.3. Problem statement and method overview

These definitions allow us to formalize the transition state clustering problem.

Problem 1 (Transition state clustering)**.** *Given a set of regular demonstrations D and transition identification function* **T***, find a segmentation* **S**.

Candidate transitions: To implement **T**, TSC fits a Gaussian mixture model (GMM) to sliding windows over each of the demonstration trajectories and identifies consecutive times with different most-likely mixture components.

Transition state clusters: The states at which those transitions occur are called transition states. TSC uses a GMM to cluster the transition states in terms of spatial and temporal similarity to find **S**.

Optimizations: To avoid setting the number of clusters at each level of the hierarchy in advance, the number of regions are determined by a DP prior. A series of merging and pruning steps remove sparse clusters and repetitive loops.

4. Transition state clustering

In this section, we describe how to take a set of given transition states Γ, and cluster them across demonstrations. The next section describes one approach to automatically generate Γ.

4.1. Non-parametric mixture models

Hyper-parameter selection is a known problem in mixture models. Recent results in Bayesian statistics can mitigate some of these problems by defining a soft prior of the number of mixtures. Consider the process of drawing samples

Algorithm 1: Transition State Clustering

```
1: Input: $\Gamma$ transition states, $\rho$ pruning parameter
2: Fit a mixture model to the set of transition states $\Gamma$ in the kinematic states.
3: Fit a mixture model to the sensory features for transitions within every kinematic cluster $i$.
4: Fit a mixture model to the times from every kinematic and sensory cluster pair $(i,j)$.
5: Remove clusters that contain fewer than transition states from fewer than $\rho \cdot N$ distinct demonstrations.
6: Output: A set of transitions, which are regions of the state-space and temporal intervals defined by Gaussian
   confidence intervals.
```

from a GMM. We first sample some c from a categorical distribution, one that takes on values from $(1 \dots m)$, with probabilities ϕ, where ϕ is a m dimensional simplex:

$$c \sim cat(m, \phi)$$

Then, conditioned on the event $\{c = i\}$, we sample from a multivariate Gaussian distribution:

$$\mathbf{x}_i \sim N(\boldsymbol{\mu}_i, \Sigma_i)$$

We can see that sampling a GMM is a two-stage process of first sampling from the categorical distribution and then conditioning on that sample.

The key insight of Bayesian non-parametrics is to add another level (or multiple levels) to this model. The DP defines a distribution over discrete distributions; in other words, a categorical distribution with certain probabilities and setting of m itself is a sample from a DP (Kulis and Jordan, 2012). To sample from the DP-GMM, one must first sample from the DP, then sample from the categorical distribution, and finally sample from the Gaussian:

$$(m, \phi) \sim DP(H, \alpha), \quad c \sim cat(m, \phi), \quad \mathbf{x} \sim N(\boldsymbol{\mu}_i, \Sigma_i)$$

The parameters of this model can be inferred with variational expectation maximization (see Appendix A). We use this model in TSC at each layer of the hierarchical clustering. For each layer, the inference procedure is independent.

4.2. Clustering algorithm

We now present the clustering algorithm, which is summarized in Algorithm 1. In a first pass, the transition states are clustered with respect to the kinematic states, then sub-clustered with respect to the sensory states, and then, we temporally sub-cluster. The sub-clusters can be used to formulate the segmentation criteria.

Kinematic step: We want our model to capture that transitions that occur in similar positions in the state-space across all demonstrations are actual transitions, and we would like to aggregate these transitions into logical events. Hypothetically, if we had infinite demonstrations, Γ would define a density of transition events throughout the state space. The modes of the density that intuitively represent a propensity of a state x to trigger a segment change are of key interest to us.

We can think of the set of identified transition states Γ as a sample of this density. We fit a DP-GMM to kinematic features of the transition states. Each transition state will have an assignment probability to one of the mixture components. We convert this to a hard assignment by assigning the transition state to the most likely component.

Sensory step: Then, we apply the second level of DP-GMM fitting over the sensory features (if available). Within each kinematic cluster, we fit a DP-GMM to find sub-clusters in the sensory features. Note that the transitions were only identified with kinematic features. This step grounds the detected transitions in sensory clusters.

Temporal step: Finally, we apply the last level of DP-GMM fitting over the time axis. Without temporal localization, the transitions may be ambiguous. For example, in a figure-of-eight motion, the robot may pass over a point twice in the same task. Conditioned on the particular state-space cluster assignment, we can fit a DP-GMM each to each subset of times. The final result contains sub-clusters that are indexed both in the state space and in time.

Enforcing consistency: The learned clusters will vary in size as some may consist of transitions that appear only in a few demonstrations. The goal of TSC is to identify those clusters that correspond to state and time transition conditions common to all demonstrations of a task. We frame this as a pruning problem, where we want to enforce that all learned clusters contain transitions from a fraction of ρ distinct demonstrations. Clusters whose constituent transition states come from fewer than a fraction of ρ demonstrations are *pruned*. Here ρ should be set based on the expected rarity of outliers. For example, if ρ is 100%, then the only mixture components that are found are those with at least one transition state from every demonstration (i.e. the regularity assumption). If ρ is less than 100%, then it means that every mixture component must cover some subset of the demonstrations. In our experiments, we set the parameter ρ to 80% and show the results with and without this step.

Segmentation criteria: Finally, if there are k remaining clusters $\{C_1, \dots, C_k\}$, we can use these clusters to form a criteria for segmentation. Each cluster is formed using a GMM triplet in the kinematic state, visual state, and time.

The quantiles of the three GMMs will define an ordered sequence of regions $[\rho_1, \ldots, \rho_k]$ over the state space and each of these regions has an associated time interval defined by the Gaussian confidence interval for some confidence level z_α.

5. Algorithm analysis

TSC is a hierarchical clustering algorithm that groups together transition events with similar time, kinematic, and sensory features. Next, we analyze some of the algorithm design choices and interpretations of the different steps.

5.1. *Hierarchical clustering interpretation*

TSC is a form of top-down divisive hierarchical clustering. Given a set of transitions, it groups those transitions based on kinematics, sensory input, and temporal conditions. The hierarchical clustering allows this clustering to be *jagged*, where parent clusters can have varying sizes of sub-clusters. One design choice was to treat the time and state axes in two different clustering steps, while other authors have proposed augmenting the state space with time (Lee et al., 2015). We empirically found that the scaling differences between the temporal features and the state features made augmenting the state space with time difficult to tune in practice. A hierarchical approach ensures that logically grouped features are clustered together.

5.2. *Probabilistic model*

Kinematics only: It can also be viewed probabilistically. Let us first consider a single layer of the hierarchy, where transitions are only clustered in the kinematic state. Over all of the demonstrations D, there is a corresponding set Γ of all transition states. We model the set Γ as samples from an underlying parametrized distribution over the state space $x \in \mathbf{R}^p$:

$$\Gamma \sim f_\theta(x)$$

Then, we can model the distribution as a GMM:

$$f_\theta(x) = GMM(\pi, \{\mu_1, \ldots, \mu_k\}, \{\Sigma_1, \ldots, \Sigma_k\})$$

The interpretation of this distribution is π describes the fraction of transitions assigned to each mixture component, μ_i describes the centroid of the mixture component, and Σ_i describes the covariance. One can think of these as defining ellipsoids in the state-space that characterize regions where transitions occur.

Multiple layers: With multiple layers of clustering hierarchy the probabilistic interpretation is a bit more complicated. However, it can be seen defining a GMM conditioned on the parent layer's assignment to a mixture component. Suppose, we have both kinematic and visual states:

$$\Gamma \sim f_\theta(x, v)$$

Using the chain rule, we can decompose $f_\theta(x, v)$ into two independently parametrized densities p, q:

$$f_\theta(x, t) = p_{\theta_p}(x) \cdot q_{\theta_q}(v \mid x)$$

We can model both p and q as mixture models, and with the simplifying assumption that the state-space mixture component is a sufficient statistic for q_{θ_q}, the hierarchical clustering process is a hard-assignment version of this inference problem.

6. Gaussian mixture transition identification

Although we can use any transition identification function to obtain Γ (as long as it satisfies the assumptions), we present one implementation based of Gaussian mixtures that we used in a number of our experiments. We found that this GMM approach was scalable (in terms of data and dimensionality) and had fewer hyper-parameters to tune than more complex models. Combined with the subsequent hierarchical clustering, this approach proved to be robust in all of our experiments.

6.1. *Transition identification algorithm*

Each demonstration trajectory d_i is a trajectory of T_i state vectors $[x_1, \ldots, x_{T_i}]$. For a given time t, we can define a window of length ℓ as

$$\mathbf{w}_t^{(\ell)} = [x_{t-\ell}, \ldots, x_t]^\intercal$$

Then, for each demonstration trajectory we can also generate a trajectory of $T_i - \ell$ windowed states:

$$\mathbf{d}_i^{(\ell)} = [\mathbf{w}_\ell^{(\ell)}, \ldots, \mathbf{w}_{T_i}^{(\ell)}]$$

Over the entire set of windowed demonstrations, we collect a dataset W of all of the $\mathbf{w}_t^{(\ell)}$ vectors. We fit a DP-GMM to these vectors. This model defines m multivariate Gaussian distributions and a probability that each observation $\mathbf{w}_t^{(\ell)}$ is sampled from each of the m distributions. We annotate each observation with the most likely mixture component. Times such that $\mathbf{w}_t^{(\ell)}$ and $\mathbf{w}_{t+1}^{(\ell)}$ have different most likely components are marked as transitions. This algorithm is summarized in Algorithm 2. For each demonstration it returns a candidate set of transitions. Which can be used to construct the set of transition states Γ.

6.2. *Capturing dynamics through windowing*

The role of windowing is to capture some information about the dynamics of the trajectory. Consider the case when $\ell = 1$, this means that W is just a dataset of all of the states observed over all trajectories. Fitting a GMM to this data, identifies clusters in the distribution of states visited by the demonstrations.

Algorithm 2: Transition Identification

```
1: Input: $D$ demonstrations, $\ell$ a window size, and $\alpha$ a DP prior.
2: For each demonstration, generate a set of sliding windows of $\mathbf{w}_t^{(\ell)} = [\mathbf{x}_{t-\ell}, \ldots, \mathbf{x}_t]^\intercal$. Let $W$ be the set of all sliding
   windows across all demonstrations.
3: Fit a mixture model to $W$ assigning each state to its most likely component.
4: Identify times $t$ in each demonstration when $\mathbf{w}_t$ has a different most likely mixture component than $\mathbf{w}_{t+1}$, start and
   finish times ($t = 0$, $t = T_i$) are automatically transitions.
5: Return: A set of transition states $\Gamma$, the $(x, v, t)$ tuples at which transitions occur.
```

Windowing ($\ell > 1$) finds clusters in the distribution of sequences of states. To interpret what this means, we highlight a well-known result from Bayesian statistics. Ghahramani and Jordan (1993) explained how GMMs are equivalent to Bayesian local linear regression, where each mixture component defines a regression surface. With this interpretation, a windowed GMM can be interpreted as finding the windows that not only lie near the same states but also lie on the same regression surface, i.e. similar local linear transition laws.

For intuition, consider the following discrete-time dynamical system:

$$x_{t+1} = \xi(x_t)$$

Suppose, we want to model the behavior of this system around a state μ. Let x_t be drawn from a Gaussian distribution $x_t \sim N(\mu, \Sigma)$, and measure how this Gaussian distribution propagates through the system. This models the distribution of states that will be observed around the neighborhood of $x_t = \mu$.

If ξ is linear, then each pair of states x_t, x_{t+1} are related by

$$x_{t+1} = Ax_t$$

It follows that the vector $\binom{x_t}{x_{t+1}}$ is a multivariate Gaussian centered at $\binom{\mu}{A\mu}$. Then, the problem of learning ξ in this neighborhood (i.e. recovering the local dynamics A) reduces to finding the parameters of this Gaussian distribution. For multivariate Gaussians, the conditional expectation over any projection is a linear estimate, and we can see that it is equivalent to regression:

$$\arg\min_{A} \sum_{t=1}^{T-1} \|Ax_t - x_{t+1}\| = \mathbf{E}[x_{t+1} \mid x_t]$$

A natural extension is a ξ that is piecewise-linear and can be modeled as switched linear dynamical system. That is, there exists m $d \times d$ matrices $\{A^{(1)}, \ldots, A^{(m)}\}$:

$$\mathbf{x}_{t+1} = A_t\mathbf{x}_t + \mathbf{w}_t : A_t \in \{A^{(1)}, \ldots, A^{(m)}\} \tag{1}$$

Assuming that each of these m linear regimes is active in a small neighborhood centered at $\{\mu_1, \ldots, \mu_m\}$, respectively, the Gaussian model extends to a GMM with m components.

7. Loop compaction

Next, we describe our approach to make the model resilient to noise in the form of loops.

7.1. "Loops" in surgical demonstrations

Loops are common in surgical demonstrations. For example, a surgeon may attempt to insert a needle two or three times before success. These looping actions will occur a varying number of times in each demonstration leading to temporal variability. This challenge is not merely theoretical, and practical datasets often contain a significant number of these looping motions. For example, the 30 Knot Tying demonstrations from the JIGSAWS dataset contain 25 looping motions (Gao et al., 2014). To be able to extract a consistent segmentation criteria, we must "compact" these loops into a single logical motion. We apply this step after Algorithm 1 (transition identification).

7.2. Compaction algorithm

The key question is how to differentiate between repetitions that are part of the demonstration and those that correspond to looping actions. The sequence might contain repetitions not due to looping. As a heuristic, we threshold the L2 distance between consecutive segments with repeated transitions. If the L2 distance is low, we know that the consecutive segments are happening in a similar location as well. In our datasets, this is a good indication of looping behavior.

For each demonstration, we define a segment $\mathbf{s}^{(j)}[t]$ of states between each transition state. The challenge is that $\mathbf{s}^{(j)}[t]$ and $\mathbf{s}^{(j+1)}[t]$ may have a different number of observations and may be at different time scales. To address this challenge, we apply dynamic time warping (DTW). Since segments are locally similar up to small time variations, DTW can find a most-likely time alignment of the two segments.

Let $\mathbf{s}^{(j+1)}[t^*]$ be a time aligned (with respect to $\mathbf{s}^{(j)}$) version of $\mathbf{s}^{(j+1)}$. Then, after alignment, we define the L_2 metric between the two segments:

$$d(j, j+1) = \frac{1}{T}\sum_{t=0}^{T}(\mathbf{s}^{(j)}[i] - \mathbf{s}^{(j+1)}[i^*])^2$$

When $d \leq \delta$, we compact two consecutive segments. Here δ is chosen empirically and a larger δ leads to a sparser distribution of transition states, and a smaller δ leads to more transition states. For our needle passing and suturing experiments, we set δ to correspond to the distance between two suture/needle insertion points; thus, differentiating between repetitions at the same point versus at others. However, since we are removing points from a time series, from every following observation, we shift the time stamp back by the length of the compacted segments.

8. Results

We evaluate TSC's robustness in the following way.

1. *Precision.* Results suggest that TSC significantly reduces the number of false-positive segments in simulated examples with noise.
2. *Recall.* Among algorithms that use piecewise linear segment models, results suggest TSC recovers segments of a generated piecewise linear trajectory more consistently in the presence of process and observation noise.
3. *Applicability to real-world data.* Result suggest that TSC recovers qualitatively relevant segments in real surgical trajectory data.

8.1. Precision in synthetic examples

Our first experiment evaluates the following hypothesis: TSC significantly reduces the number of false-positive segments in a simple simulated example with noise. These experiments evaluate TSC against algorithms with a single level of clustering.

We now provide a comparison of seven alternative segmentation criteria.

1. *Zero-Velocity Crossing (VEL):* This algorithm detects a change in the sign of the velocity.
2. *Smoothed Zero-Velocity Crossing (VELS):* This algorithm applies a low-pass filter (exponentially weighted moving average) to the trajectory, and then detects a change in the sign of the velocity.
3. *Acceleration (ACC):* This algorithm detects any change in the velocity by looking for non-zero acceleration.
4. *Gaussian Mixture Model (GMM):* This algorithm applies a GMM model to the observed states and detects changes in most likely assignment. The number of clusters was set to two.
5. *Windowed Gaussian Mixture Model (GMMW):* This algorithm is the first phase of TSC. It applies a GMM to windows of size two, and detects changes in most likely assignment. The number of clusters was set to two, unlike in TSC where we use the DP to set the number of clusters.
6. *Auto-Regressive Mixture (AR):* This model fits a piecewise linear transition law to the observed data.
7. *Coresets (CORE):* We evaluate against a standard coreset model (Sung et al., 2012; Volkov et al., 2015), and the particular variant is implemented with weighted k-means. We applied this to the same augmented state vector as in the previously mentioned GMM.
8. *Transition state clustering (TSC):* Our proposed approach with a pruning threshold of 0.8 and no loop compaction.

8.1.1. Bouncing ball. We first revisit the example in the introduction of the bouncing ball, which can be modeled as the following 1D double-integrator system:

$$\ddot{x} = -9.8\,{}^{\mathrm{m}}\!/\mathrm{s}^2$$

This system is observed with additive Gaussian white noise with std 10 (moderate noise):

$$y = x + N(0, 10)$$

and std 20 (high noise):

$$y = x + N(0, 20)$$

The system is initialized at $x_0 = 122.5$ and bounces when $x = 20$, at which point the velocity is negated. Figure 3 illustrates the ideal trajectory and noisy realizations of these trajectories.

We apply the segmentation algorithms to the trajectories and plot the results in Figure 4. When there is no noise, all of the algorithms are equally precise, and there is no trouble with corresponding segments across demonstrations. All of the “rate-of-change” methods (VEL, VELS, ACC) reliably identify the point where the ball bounces. The GMM and the CORE methods do not segment the trajectory at the bounce point. On the other hand, the windowed GMM takes two consecutive positions and velocities into account during the clustering. Similarly, the autoregressive model can accurately identify the bounce point. With no noise, TSC has little difference with the windowed GMM.

Differences arise when we observe the trajectory with additive Gaussian noise. The “rate-of-change” methods have some spurious segmentation points due to noise. The GMM-based methods are more robust to this noise, and they retain similar precision. This motivates our choice of the first phase of the TSC algorithm using a windowed GMM approach. However, the GMM approaches still have some spurious transitions. With these spurious points, it becomes challenging to reliably correspond trajectories across segments. Thus, TSC applies a second phase of clustering to correspond the transitions and prune the sparse clusters. This results in accurate segmentation even in the presence of noise.

As the noise increases, TSC is still able to find accurate segments. In the high-noise case, the bounce point is still identified in four out of five trajectories. It is important to note that we do not claim that one segmentation

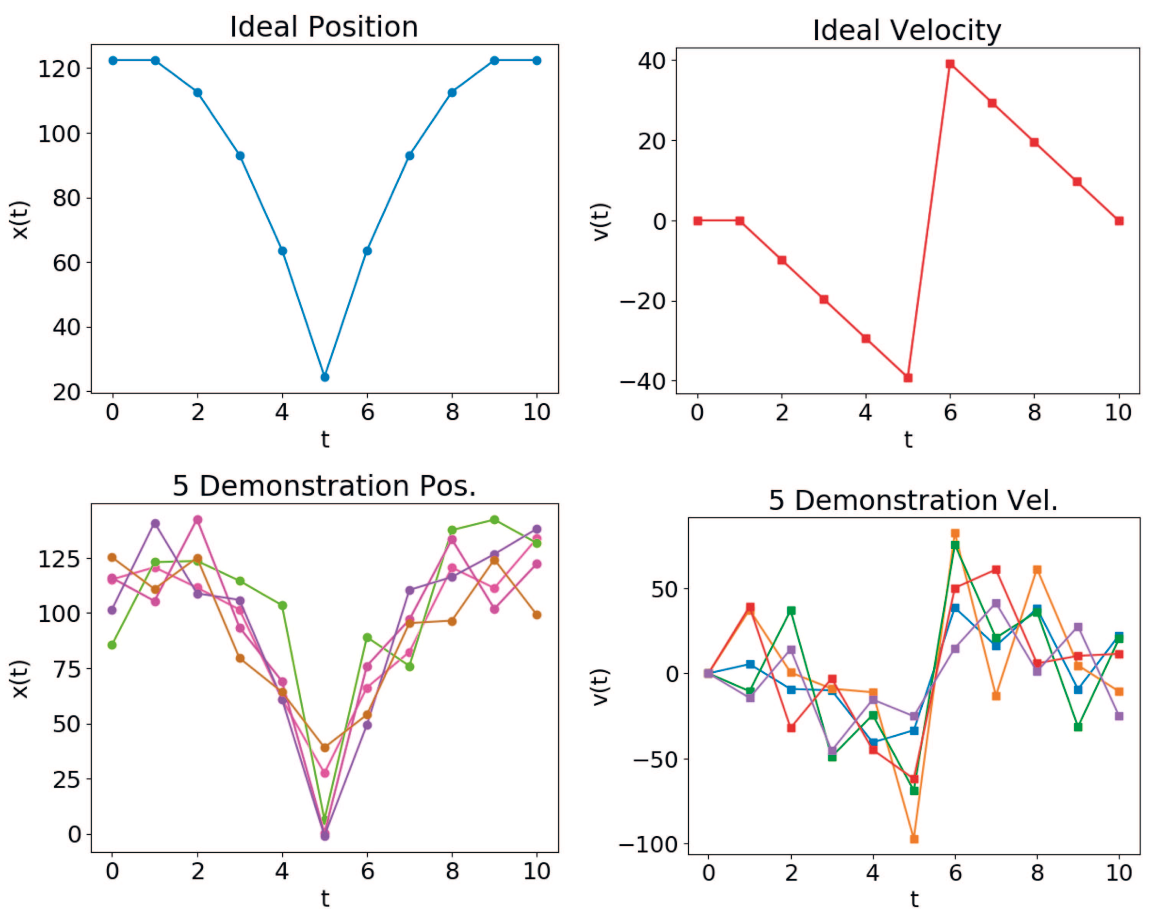


Fig. 3. (Top) Position and velocity of the bouncing ball without noise. (Bottom) Five trajectories of the ball with different realizations of the noise.

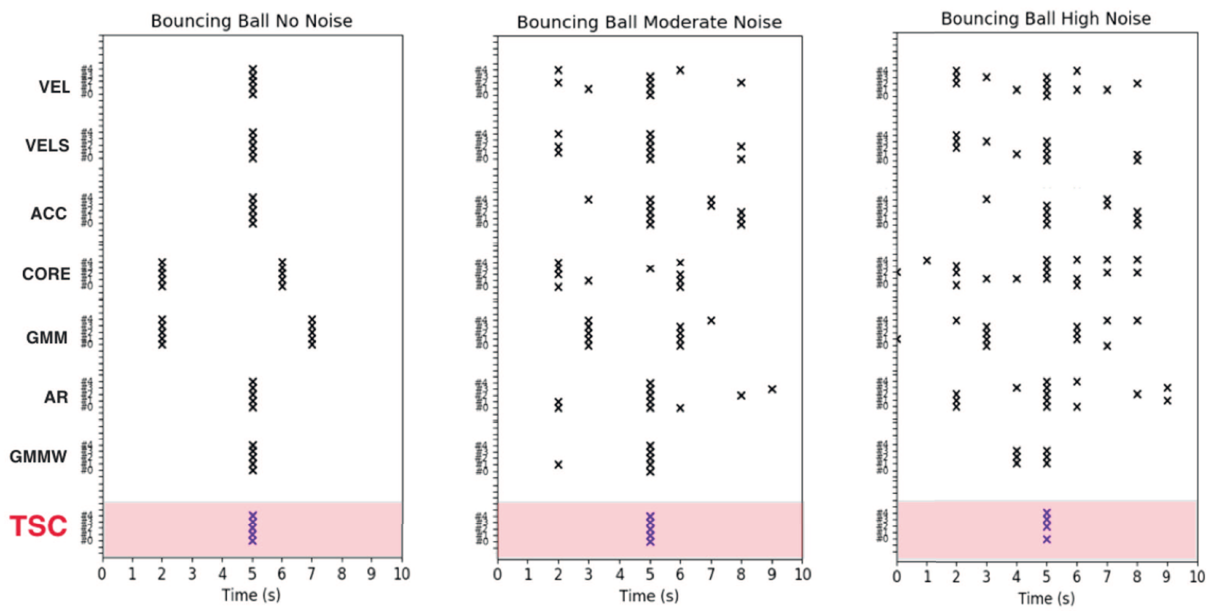


Fig. 4. Plots of the identified transitions with each segmentation algorithm with and without noise. Whereas all techniques are precise when there is no noise, TSC is the most robust in the presence of noise.

algorithm is more accurate than another, or that TSC more accurately reflects "real" segments. These results only suggest that TSC is more precise than alternatives; that is, given the assumptions in TSC it consistently recovers segments according to those assumptions. The next experiments will study the recall characteristics.

8.1.2. Bouncing ball with air resistance. In the first set of experiments, we illustrate TSC's robustness to variance in the state space. Next, we illustrate how TSC can still correspond segments with temporal variation. Consider the dynamics of the bouncing ball with an term to account for air resistance:

$$\ddot{x} = -9.8\,{}^{\mathrm{m}}\!/\!{}_{\mathrm{s}^2} + K_v\dot{x}$$

We draw the air-resistance constant K_v uniformly from $K_v \sim U[1,5]$. The consequence is that the ball will bounce at different times in different trajectories.

Figure 5 illustrates the results. In the five trajectories, the ball bounces between time steps 5 and 7. With no noise VEL, VELS, ACC, GMMW, and TSC can identify the bounce point. Then, the system is observed with additive Gaussian white noise with std 10:

$$y = x + N(0, 10)$$

We find that TSC recovers a consistent set of segments even with the temporal variation.

8.1.3. Hybrid approaches. In the previous experiments, we presented TSC using a windowed GMM approach to identify transitions. Next, we consider TSC with alternative transition identification functions. Consider a "figure-of-eight" trajectory defined parametrically as

$$x = \cos(t)$$

$$y = 0.5\sin(2t)$$

The trajectory is visualized in Figure 6. The trajectory starts at the far right and progresses until it returns to the same spot. Velocity-based segmentation finds one transition point where there is a change in direction (far left of the trajectory; Figure 7). A windowed GMM where the number of clusters is set by a DP finds three transition points. These three points correspond to the far left point as well as the crossing point in the figure-of-eight (happens twice). These are two different segmentation criteria, and both are reasonable with respect to their respective assumptions.

Next, this parametric trajectory is observed with additive Gaussian noise of std 0.1 (Figure 6). We see that both the GMM approach and the velocity approach have several spurious transitions (Figure 7). TSC can improve the precision of both techniques by adding a layer of clustering.

8.2. Rotations

Handling orientations is a challenging problem due to the topology of $SO(3)$ (Ude et al., 2014). As an example of what can go wrong consider a 2D square rotating in the plane. We construct a 1×1 m^2 2D square and track a point on the corner of the 2D square. The 2D square rotates clockwise in $\frac{\pi}{10}$ rad/s for 10 time steps, then switches, and rotates the other direction at the same angular speed. The state of the system is the (x, y) position of the corner. We add 0.1 m standard deviation Gaussian observation noise to the observed trajectories.

We apply the segmentation algorithms to five trajectories and plot the results in Figure 8. As before, with no noise, all of the techniques are equally precise. In this example, there is a difference between how the different techniques segment the trajectories. The rate-of-change methods segment the trajectory at the point when the block changes rotation direction. The GMM and the windowed GMM approaches cuts the trajectory into three even segments, missing the direction change. TSC cuts the trajectory into four segments including the direction change. TSC differs from the windowed GMM because it sets the number of clusters using the DP prior. With noise, the rate-of-change techniques have a number of spurious segments. The GMM-based approaches are more robust and TSC improves the windowed GMM even further by clustering the detected transitions. However, if the initial transitions were found in angular space, then TSC would have found one segment. In this sense, the definition of the state-space changes the segments found. We hope to explore these issues in more detail in future work.

8.3. Recall in synthetic examples

Comparing different segmentation models can be challenging due to differing segmentation criteria. However, we identified some algorithms that identify locally linear or near-linear segments. We developed a synthetic dataset generator to generate piecewise linear segments and compared the algorithms on the generated dataset. Note, we do not intend this to be a comprehensive evaluation of the accuracy of the different techniques, but more a characterization of the approaches on a locally linear example to study the key tradeoffs.

8.3.1. Overview. We model the trajectory of a point robot with two-dimensional position state (x, y) between k goal points $\{g_1, \ldots, g_k\}$. We apply position control to guide the robot to the targets and without disturbance, this motion is linear (Figure 9(a)). We add various types of disturbances (and in varying amounts) including Gaussian observation noise, low-frequency process noise, and repetitive loops (Figure 9(b)–(d)). We report noise values in terms of standard deviations. Figure 10 illustrates the relative magnitudes. A demonstration d_i is a sample from the following system.

Task: Every segmentation algorithm will be evaluated in its ability to identify the $k - 1$ segments (i.e. the paths

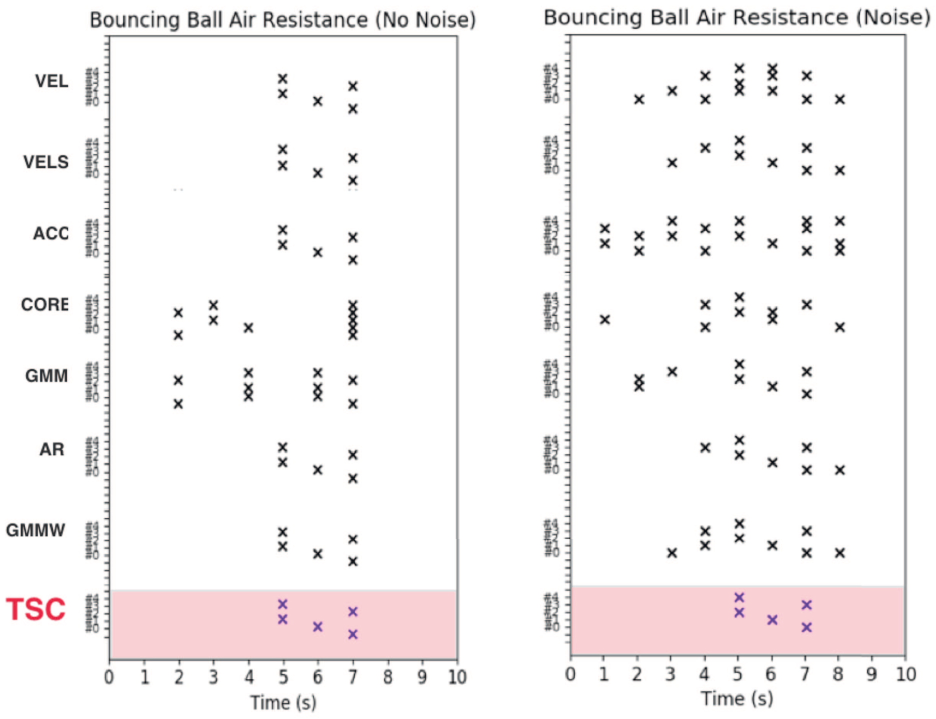


Fig. 5. Plots of the identified transitions with each segmentation algorithm with and without noise. In this example, temporal variation is added by incorporating a random "air resistance" factor. TSC is consistent even in the presence of this temporal variation.

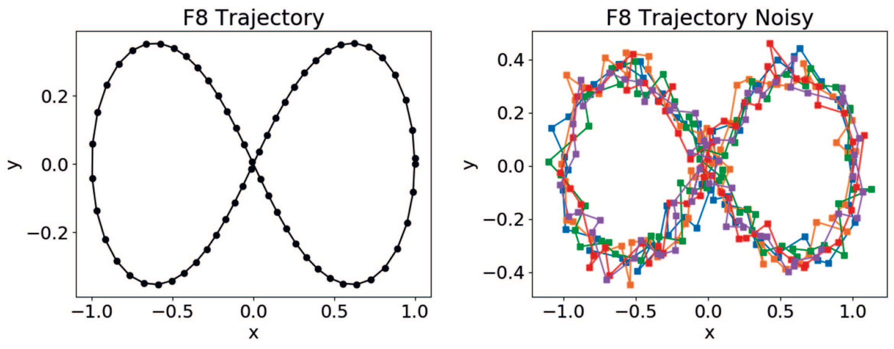


Fig. 6. A "figure-of-eight" trajectory in the plane and five noisy demonstrations. The trajectory starts at the far right and progresses until it returns to the same spot.

between the goal points). Furthermore, we evaluate algorithms on random instances of this task. In the beginning, we select three random goal points. From a fixed initial position, we control the simulated point robot to the goal points with position control. Without any disturbance, this follows a linear motion. For a given noise setting, we sample demonstrations from this system and apply/evaluate each algorithm. We present results aggregated over 20 such random instances. This is important since many of the segmentation algorithms proposed in literature have some crucial hyper-parameters, and we present results with a *single* choice of parameters averaged over multiple tasks. In this way, the hyper-parameter tuning cannot overfit to any given instance of the problem and has to be valid for the entire class of tasks. We believe that this is important since tuning these hyper-parameters in practice (i.e. not in simulation) is challenging since there is no ground truth. The experimental code is available at: http://berkeleyautomation.github.io/tsc/.

Five algorithms: We compare TSC against alternatives where the authors explicitly find (or approximately find) locally linear segments. It is important to reiterate that different segmentation techniques optimize different objectives, and this benchmark is meant to characterize the performance on a common task. All of the techniques are based on Gaussian distributions or linear autoregressive models.

1. *(GMM)* (same as previous experiment). In this experiment, we set the parameter to the optimal choice of three without automatic tuning.

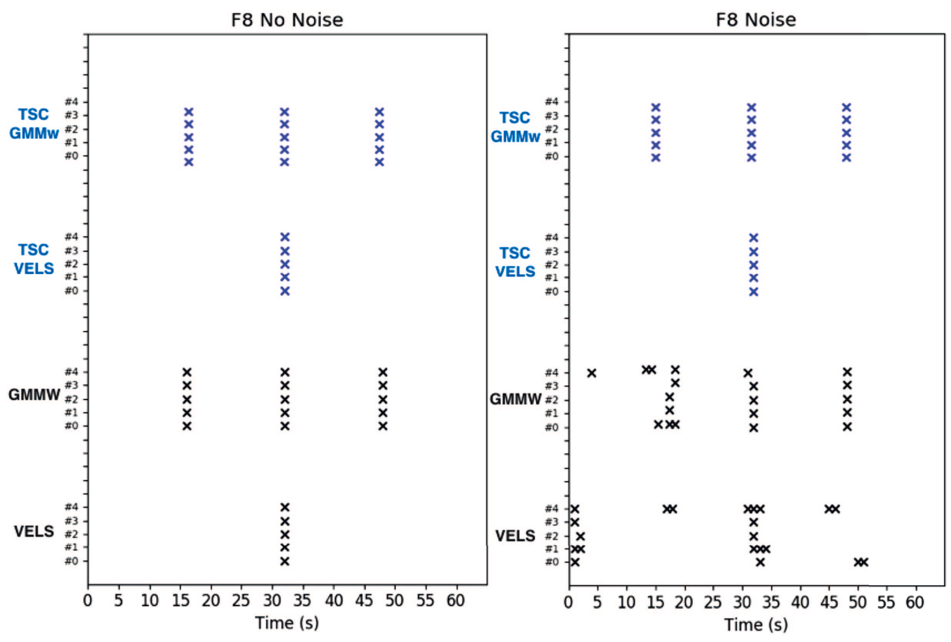


Fig. 7. Plots of the identified transitions with each segmentation algorithm with and without noise. Velocity-based segmentation finds one transition point where there is a change in direction. A windowed GMM where the number of clusters is set by a DP finds three transition points. TSC can improve the precision of both techniques.

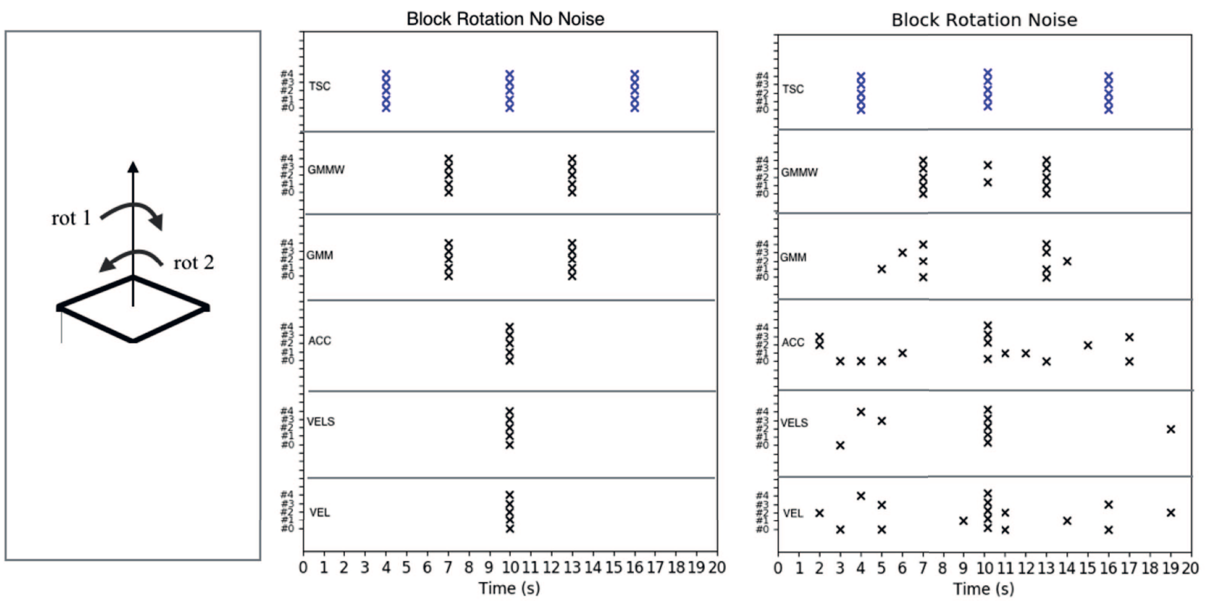


Fig. 8. Plots of the identified transitions with each segmentation algorithm with and without noise. Although all techniques are precise when there is no noise, TSC is the most robust in the presence of noise but finds additional segments.

2. *(GMM+HMM)*. A natural extension to this model is to enforce a transition structure on the regimes with a latent Markov chain (Asfour et al., 2008; Calinon and Billard, 2004; Kruger et al., 2010; Vakanski et al., 2012). We use the same state vector as above, without time augmentation as this is handled by the HMM. We fit the model using the forward–backward algorithm.
3. *Coresets* (same as previous experiment).
4. *HSMM*. We evaluated a Gaussian hidden semi-Markov model as used by Tanwani and Calinon (2016). We applied this model directly to the demonstrations with no augmentation or normalization of features. This was implemented with the package pyhsmm. We applied this model directly to the demonstrations with no augmentation as in the GMM approaches. We ran our MCMC sampler for 10,000 iterations, discarding the first 2,500 as burn-in and thinning the chain by 15.

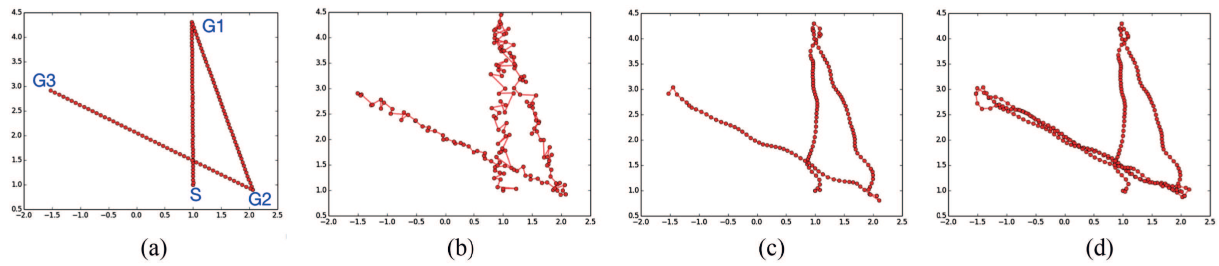


Fig. 9. One of 20 instances with random goal points $G1$, $G2$, $G3$: (a) observations from a simulated demonstration with three regimes; (b) observations corrupted with Gaussian white sensor noise; (c) observations corrupted with low-frequency process noise; and (d) observations corrupted with an inserted loop. See Figure 13 for evaluation on loops.

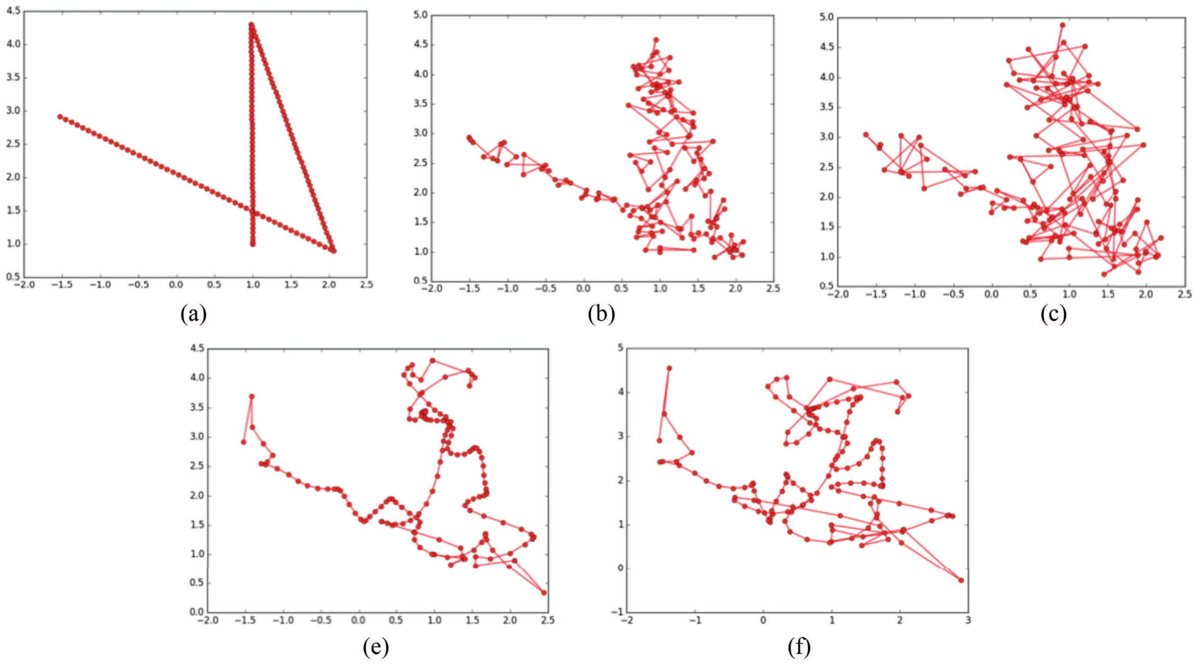


Fig. 10. (a) Nominal trajectory, (b) 1 std of high-frequency observation noise, (c) 2 std of high-frequency observation noise, (d) 1 std of low-frequency process noise, and (e) 2 std of low-frequency process noise.

5. *AR-HMM*. We evaluated a Bayesian autoregressive HMM model as used in Niekum et al. (2012). This was implemented with the packages pybasicbayes and pyhsmm-ar. The autoregressive order was 10 and we ran our MCMC sampler for 10,000 iterations, discarding the first 2,500 as burn-in and thinning the chain by 15.

Evaluation metric: There is considerable debate on metrics to evaluate the accuracy of unsupervised segmentation and activity recognition techniques, e.g. frame accuracy (Wu et al., 2015) and hamming distance (Fox et al., 2009). Typically, these metrics have two steps: (1) segments to ground-truth correspondence; and (2) then measuring the similarity between corresponding segments. We have made this feature extensible and evaluated some different accuracy metrics (Jaccard similarity, frame accuracy, segment accuracy, intersection over union). We found that the following procedure led to the most insightful results, differentiating the different techniques.

In the first phase, we match segments in our predicted sequence to those in the ground truth. We do this with a procedure identical to that proposed by Wu et al. (2015). We define a bi-partite graph of predicted segments to ground-truth segments, and add weighted edges where weights represent the overlap between a predicted segment and a ground-truth segment (i.e. the recall over time-steps). Each predicted segment is matched to its highest weighted ground-truth segment. Each predicted segment is assigned to exactly one ground-truth segment, while a ground-truth segment may have none, one, or more corresponding predictions.

After establishing the correspondence between predictions and ground truth, we consider a true positive (a ground-truth segment is correctly identified) if the overlap (intersection-over-union) between the ground-truth segment and its corresponding predicted segments is more

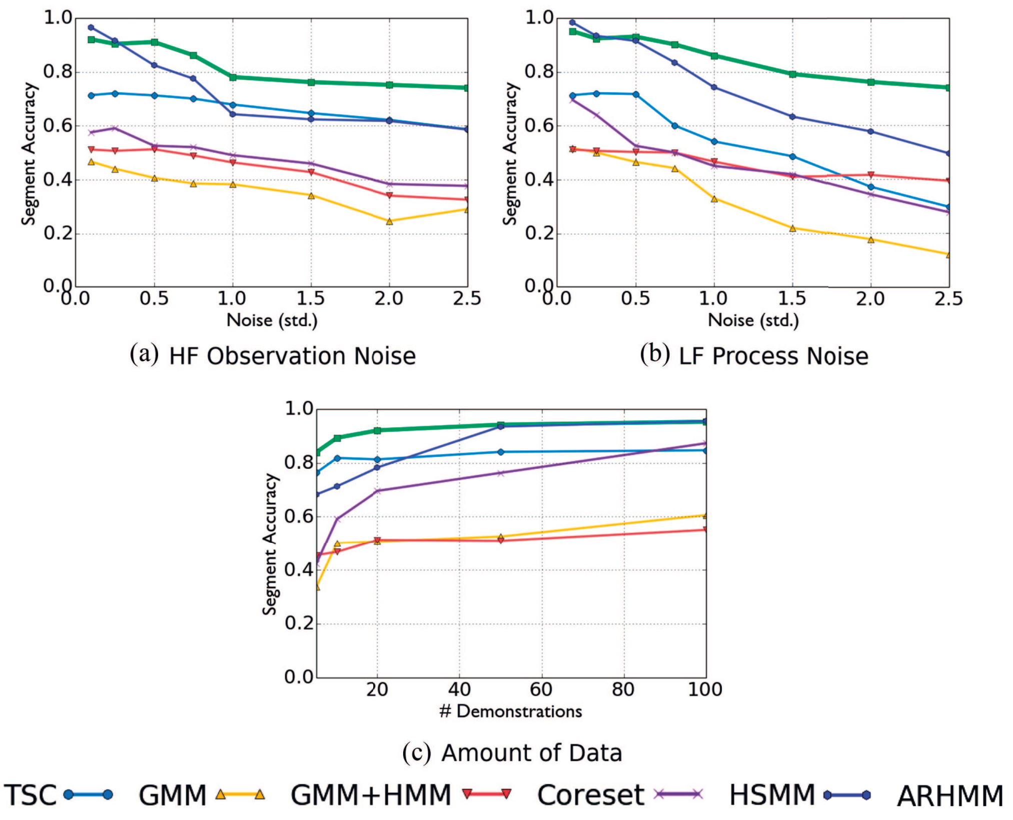


Fig. 11. Each data point represents 20 random instances of a three-segment problem with varying levels of high-frequency noise, low-frequency noise, and demonstrations. We measure the segmentation accuracy for the compared approaches. (a) TSC finds a more accurate segmentation than all of the alternatives even under significant high-frequency observation noise, (b) TSC is more robust to low-frequency process noise than the alternatives, (c) the Bayesian techniques solved with MCMC (ARHMM, HSMM) are more sensitive to the number of demonstrations provided than the others.

than a default threshold 60%. Then, we compute segment accuracy as the ratio of the ground-truth segments that are detected correctly. Wu et al. (2015) used a 40% threshold but apply the metric to real data. Since this is a synthetic example, we increase this threshold to 60%, which we empirically found accounted for boundary effects, especially in the Bayesian approaches (i.e. repeated transitions around segment endpoints).

8.3.2. Accuracy versus noise. In our first experiment, we measured the segment accuracy for each of the algorithms for 50 demonstrations. We also varied the amount of process and observation noise in the system. As Figure 10 illustrates, this is a very significant amount of noise in the data, and successful techniques must exploit the structure in multiple demonstrations. Figure 11(a) illustrates the performance of each of the techniques as a function of high-frequency observation noise. Results suggest that TSC is more robust to noise than the alternatives (nearly 20% more accurate for 2.5 std of noise). The Bayesian ARHMM approach is nearly identical to TSC when the noise is low but quickly loses accuracy as more noise is added. We

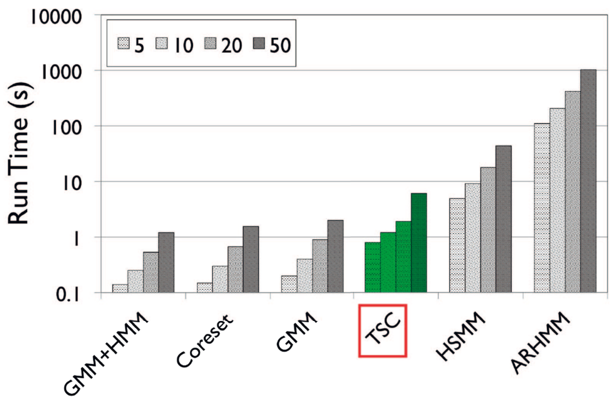


Fig. 12. TSC is about six time slower than using Coresets or the direct GMM approach, but it is over 100× faster than the MCMC for the ARHMM model.

attribute this robustness to the TSC's pruning step which ensures that only transition state clusters with sufficient coverage across all demonstrations are kept. These results are even more pronounced for low-frequency process noise (Figure 11(b)). TSC is 49% more accurate than all competitors for 2.5 std of noise added. We find that the Bayesian approaches are particularly susceptible to such noise. Furthermore, Figure 11(c) shows TSC requires no more data than the alternatives to achieve such robustness. Another point to note is that TSC is solved much more efficiently than ARHMM or HSMM that require expensive MCMC samples. Although parameter inference on these models can be solved more efficiently (but approximately) with mean-field stochastic variational inference, we found that the results were not as accurate. TSC is about six times slower than using Coresets or the direct GMM approach, but it is over $100\times$ faster than the MCMC for the ARHMM model. Figure 12 compares the runtime of each of the algorithms as a function of the number of demonstrations.

8.3.3. TSC hyper-parameters. Next, we explored the dependence of the performance on the hyper-parameters for TSC. We focus on the window size and the pruning parameter. Figure 13(a) shows how varying the window size affects the performance curves. Larger window sizes can reject more low-frequency process noise. However, larger windows are also less efficient when the noise is low. Similarly, Figure 13(b) shows how increasing the pruning parameter affects the robustness to high-frequency observation noise. However, a larger pruning parameter is less efficient at low noise levels. Based on these curves, we selected ($w = 3,\ \rho = 0.3$) in our synthetic experiments.

8.3.4. Loops. Finally, we evaluated four algorithms on how well they can detect and adjust for loops. TSC compacts adjacent motions that are overly similar, while HMM-based approaches correspond similar looking motions. A HMM grammar over segments is clearly more expressive than TSC's, and we explore whether it is necessary to learn a full transition structure to compensate for loops. We compare the accuracy of the different segmentation techniques in detecting that a loop is present (Figure 14). Figure 14(a) shows that TSC is competitive with the HMM approaches as we vary the observation noise; however, the results suggest that ARHMM provides the most accurate loop detection. On the other hand, Figure 14(b) suggests that process noise has a very different effect. TSC is actually more accurate than the HMM approaches when the process noise is high, even without learning a transition structure.

8.3.5. Scaling with dimensionality. We investigate how the accuracy of TSC scales with the dimensionality of the state space. As in the previous experiments, we measured the segment accuracy for each of the algorithms for 50 demonstrations. This time we generated the line segments in increasingly higher-dimensional spaces (from 2 to 35 dimensions). The noise added to the trajectories has a std of 0.1. Figure 15(a) plots the segment accuracy as a function of the dimensionality of the state space. While the accuracy of TSC does decreases as the dimensionality increases it is more robust than some of the alternatives: ARHMM and HSMM. One possible explanation is that both of those techniques rely on Gibbs sampling for inference, which is a little more sensitive to dimensionality than the expectation-maximization inference procedure used in GMM and GMM+HMM. Figure 15(b) shows one aspect of TSC that is more sensitive to the dimensionality. The loop compaction step requires a dynamic time-warping and then a comparison to fuse repeated segments together. This step is not as robust in higher-dimensional state spaces. This is possibly due to the use of the L_2 distance metric to compare partial trajectories to compact. TSC runs in 4 seconds on the two-dimensional case, 16 seconds on the 10-dimensional case, and in 59 seconds on the 35-dimensional case.

8.4. Surgical data experiments

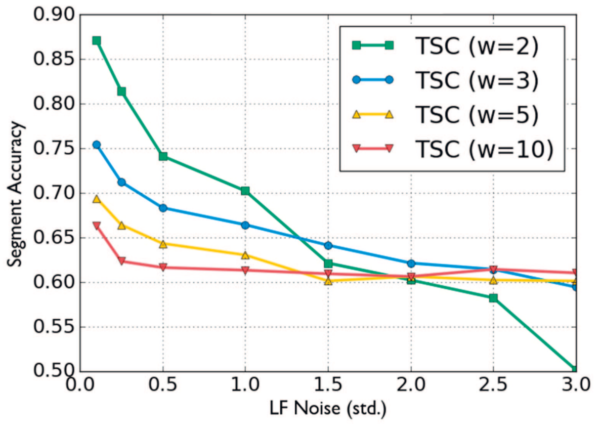
We describe the three tasks used in our evaluation and the corresponding manual segmentation (Figure 16). This will serve as ground truth when qualitatively evaluating our segmentation on real data. This set of experiments primarily evaluates the utility of segments learned by TSC. Data was collected beforehand as a part of prior work. Our hypothesis is that even though TSC is unsupervised, it identifies segments that often align with manual annotations. In all of our experiments, the pruning parameter ρ is set to 80% and the compaction heuristic δ is to 1 cm.

The state-space is the 6D end-effector position. In some experiments, we augment this state space with the following visual features.

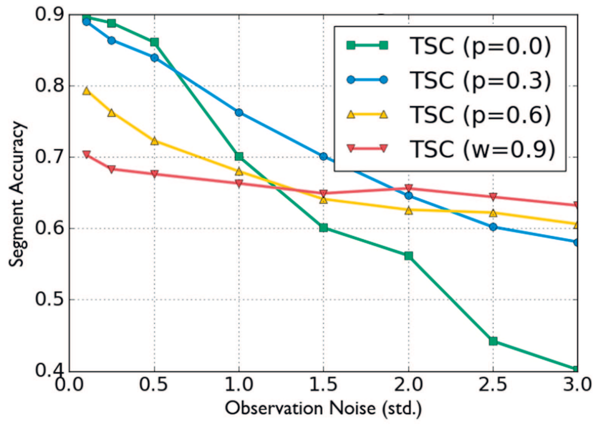
1. *Grasp*: 0 if empty; 1 otherwise.
2. *Needle penetration*. We use an estimate of the penetration depth based on the robot kinematics to encode this feature. If there is no penetration (as detected by video), the value is 0; otherwise the value of penetration is the robot's z position.

Our goal with these features was to illustrate that TSC applies to general state-spaces as well as spatial ones, and not to address the perception problem. These features were constructed via manual annotation, where the grasp and needle penetration were identified by reviewing the videos and marking the frames at which they occurred.

Circle cutting: A 5 cm diameter circle drawn on a piece of gauze. The first step is to cut a notch into the circle. The second step is to cut clockwise half-way around the circle. Next, the robot transitions to the other side cutting counter clockwise. Finally, the robot finishes the cut at the meeting point of the two cuts. As the left arm's only action is to maintain the gauze in tension, we exclude it from the analysis. In Figure 16(a), we mark six manually identified transitions points for this task from Murali et al. (2015): (1) start, (2) notch, (3) finish first cut, (4) cross-over, (5) finish second cut, and (6) connect the two cuts. For the circle cutting task, we collected 10 demonstrations by researchers who were not surgeons but familiar with operating the da Vinci Research Kit (dVRK).

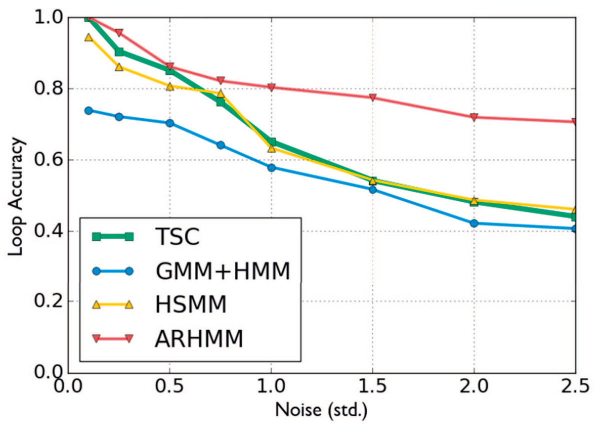


(a)

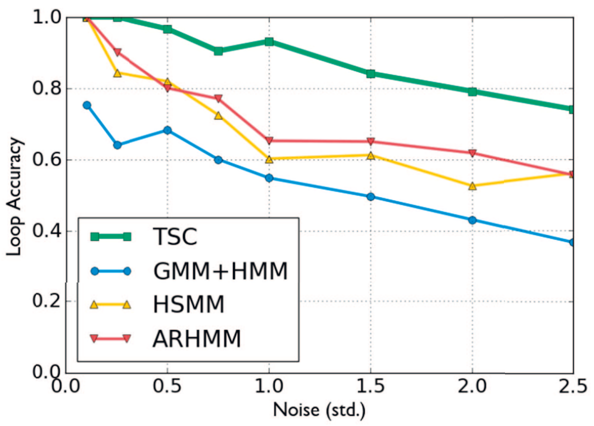


(b)

Fig. 13. (a) Performance curves of different choices of windows as a function of the process noise. Larger windows can reject higher amounts of process noise, but are less efficient at low noise levels. (b) Performance curves of different choices of the pruning threshold. Larger pruning thresholds are more robust to high amounts of observation noise but less accurate in the low-noise setting. We selected ($w = 3$, $\rho = 0.3$) in our synthetic experiments.



(a)



(b)

Fig. 14. (a) Accuracy of TSC's compaction step as a function of observation noise. TSC is competitive with the HMM-based approaches without having to model the full transition matrix. (b) TSC is actually more robust to low-frequency process noise in the loops than the HMM-based approaches.

We also perform experiments using the JIGSAWS dataset (Gao et al., 2014) consisting of surgical activity for human motion modeling. The dataset was captured using the da Vinci Surgical System from eight surgeons with different levels of skill performing five repetitions each of needle passing and suturing.

Needle passing: We applied TSC to 28 demonstrations of the needle passing task. The robot passes a needle through a hoop using its right arm, then its left arm to pull the needle through the hoop. Then, the robot hands the needle off from the left arm to the right arm. This procedure is repeated four times as illustrated with a manual segmentation in Figure 16(b).

Suturing: Next, we explored 39 examples of a 4 throw suturing task (Figure 16(c)). Using the right arm, the first step is to penetrate one of the points on the right-hand side. The next step is to force the needle through the phantom to the other side. Using the left arm, the robot pulls the needle out of the phantom and then the robot hands it off to the right arm for the next point.

8.4.1. Pruning and compaction. In Figure 19, we highlight the benefit of pruning and compaction using the suturing task as an example. First, we show the transition states without applying the compaction step to remove looping transition states (Figure 19(a)). We find that there are many more transition states at the "insert" step of the task. Compaction removes the segments that correspond to a loop of the insertions. Next, we show all of the clusters found by the first step of segmentation. The centroids of these clusters are marked in Figure 19(b). Many of these clusters are

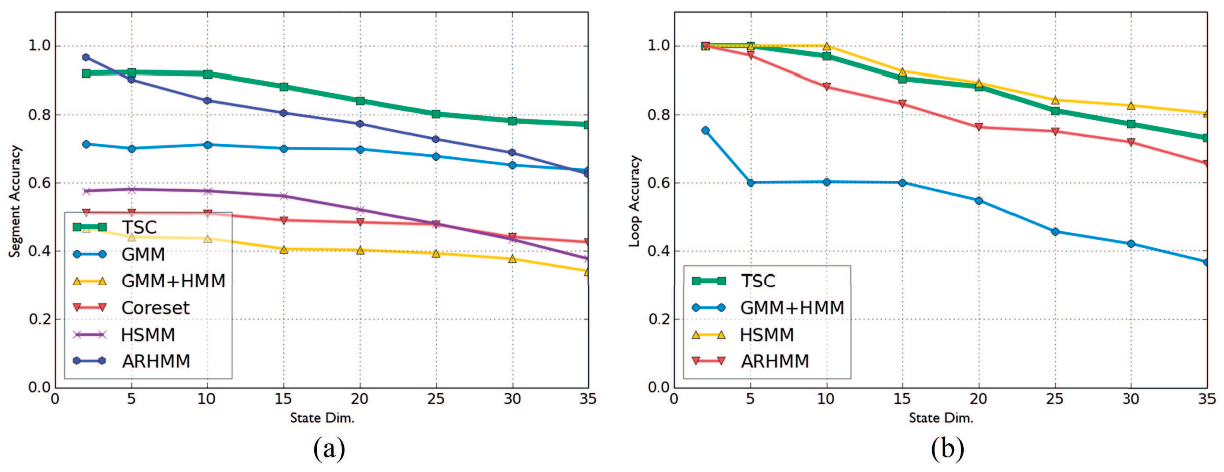


Fig. 15. We investigate how the accuracy of TSC scales with the dimensionality of the state-space. In (a) we consider the problem with no loops or compaction and in (b) we measure the accuracy of the compaction step as a function of dimensionality.

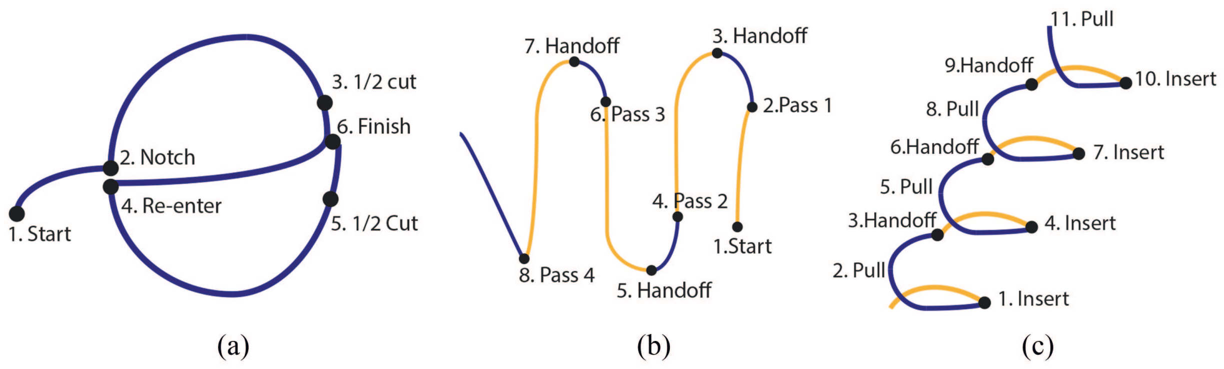


Fig. 16. Hand annotations of the three tasks: (a) circle cutting, (b) needle passing, and (c) suturing. Right arm actions are listed in dark blue and left arm actions are listed in yellow.

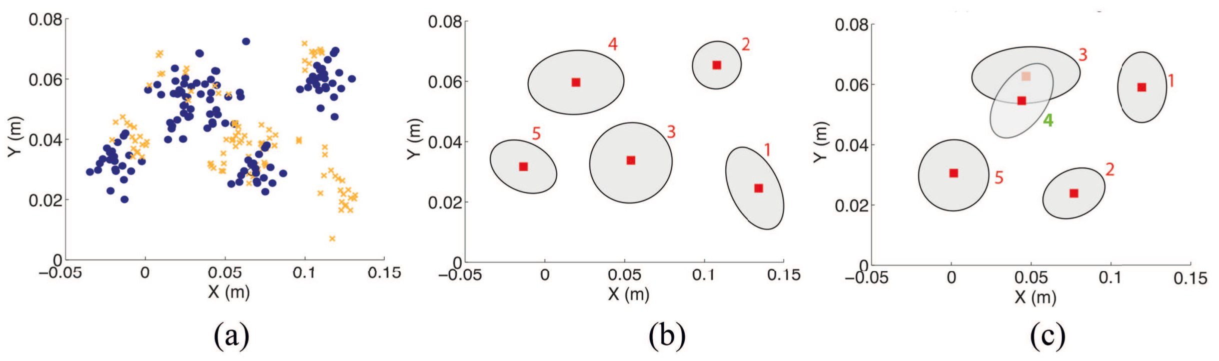


Fig. 17. (a) The transition states for the task are marked in orange (left arm) and blue (right arm). (b), (c) The TSC clusters, which are clusters of the transition states, are illustrated with their 75% confidence ellipsoid for both arms.

small containing only a few transition states. This is why we created the heuristic to prune clusters that do not have transition states from at least 80% of the demonstrations. In all, 11 clusters are pruned by this rule.

8.5. Results with surgical data

Circle cutting: Figure 20(a) shows the transition states obtained from our algorithm and Figure 20(b) shows the TSC clusters learned (numbered by time interval midpoint).

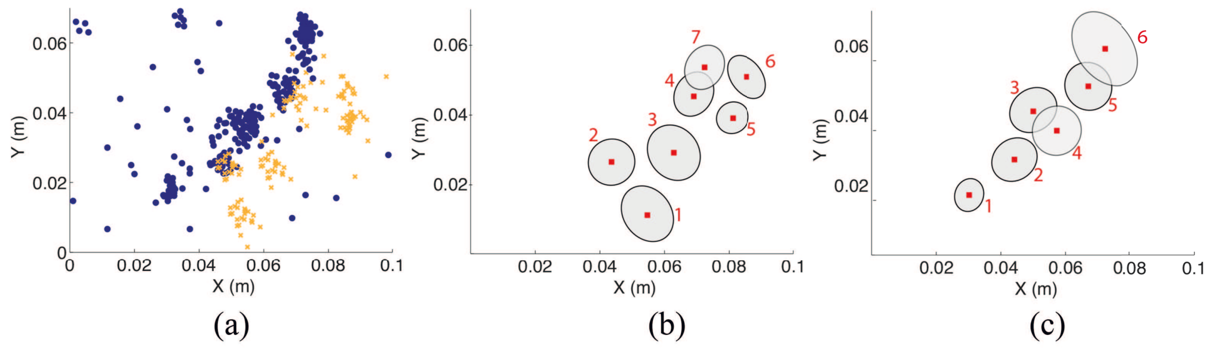


Fig. 18. (a) The transition states for the task are marked in orange (left arm) and blue (right arm). (b), (c) The clusters, which are clusters of the transition states, are illustrated with their 75% confidence ellipsoid for both arms.

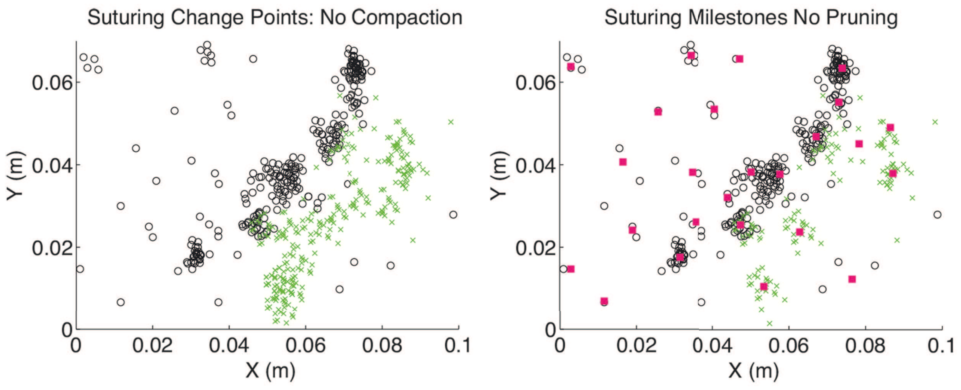


Fig. 19. We first show the transition states without compaction (in black and green), and then show the clusters without pruning (in red). Compaction sparsifies the transition states and pruning significantly reduces the number of clusters.

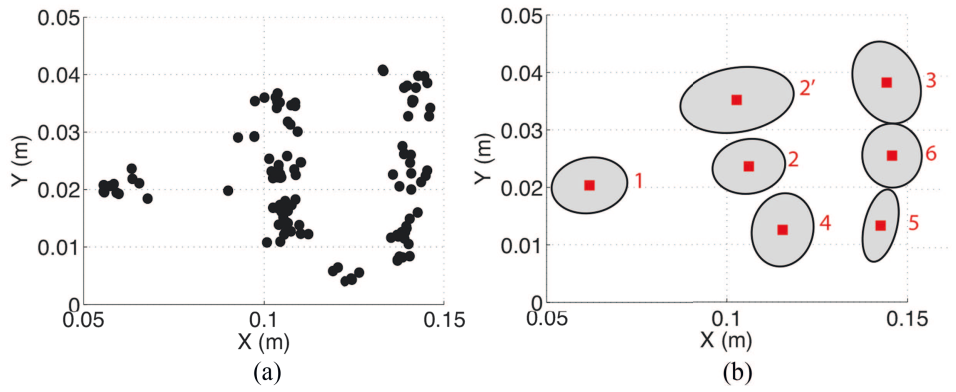


Fig. 20. (a) The transition states for the circle cutting task are marked in black. (b) The TSC clusters, which are clusters of the transition states, are illustrated with their 75% confidence ellipsoid.

The algorithm found eight clusters, one of which was pruned using our $\rho = 80\%$ threshold rule.

The remaining seven clusters correspond well to the manually identified transition points. It is worth noting that there is one extra cluster (marked 2′), that does not correspond to a transition in the manual segmentation. At 2′, the operator finishes a notch and begins to cut. While at a logical level notching and cutting are both penetration actions, they correspond to two different linear transition regimes due to the positioning of the end-effector. Thus, TSC separates them into different clusters even though the human annotators did not. This illustrates why supervised segmentation is challenging. Human annotators segment trajectories on boundaries that are hard to characterize mathematically, e.g.

Table 1. Comparison of transitions learned by TSC and transitions identified by manual annotators in the JIGSAWS dataset. We found that the transitions mostly aligned. Here 83% and 73% of transition clusters for needle passing and suturing respectively contained exactly one surgeme transition when both kinematics and vision were used. Results suggest that the hierarchical clustering is more suited to mixed video and kinematic feature spaces.

	Number of surgeme segments	Number of clusters	seg–surgeme	surgeme–seg
Needle passing TSC(Kin+Video)	14.4 ± 2.57	11	83%	74%
Needle passing TSC(Video)	14.4 ± 2.57	7	62%	69%
Needle passing TSC(Kin)	14.4 ± 2.57	16	87%	62%
Needle passing TSC(VELS)	14.4 ± 2.57	13	71%	70%
Needle passing TSC(No-H)	14.4 ± 2.57	5	28%	34%
Suturing TSC(Kin+Video)	15.9 ± 3.11	13	73%	66%
Suturing TSC(Video)	15.9 ± 3.11	4	21%	39%
Suturing TSC(Kin)	15.9 ± 3.11	13	68%	61%
Suturing TSC(VELS)	15.9 ± 3.11	17	48%	57%
Suturing TSC(No-H)	15.9 ± 3.11	9	51%	52%

is frame 34 or frame 37 the segment boundary? Supervisors may miss crucial motions that are useful for automation or learning.

Needle passing: In Figure 17(a), we plot the transition states in (x, y, z) end-effector space for both arms. We find that these transition states correspond well to the logical segments of the task (Figure 16(b)). These demonstrations are noisier than the circle cutting demonstrations, and there are more outliers. The subsequent clustering finds nine clusters (two pruned). Next, Figure 17(b) and (c) illustrate the TSC clusters. We find that again TSC learns a small parametrization for the task structure with the clusters corresponding well to the manual segments. However, in this case, the noise does lead to a spurious cluster (four marked in green). One possible explanation is that the demonstrations contain many adjustments to avoid colliding with the needle hoop and the other arm while passing the needle through leading to numerous transition states in that location.

Suturing: In Figure 18, we show the transition states and clusters for the suturing task. As before, we mark the left arm in orange and the right arm in blue. This task was far more challenging than the previous tasks as the demonstrations were inconsistent. These inconsistencies were in the way the suture is pulled after insertion (some pull to the left, some to the right, etc.), leading to transition states all over the state space. Furthermore, there were numerous demonstrations with looping behaviors for the left arm. In fact, the DP-GMM method gives us 23 clusters, 11 of which represent less than 80% of the demonstrations and thus are pruned (we illustrate the effect of the pruning in the next section). In the early stages of the task, the clusters clearly correspond to the manually segmented transitions. As the task progresses, we see that some of the later clusters do not.

8.6. Comparison with surgemes

Surgical demonstrations have an established set of primitives called surgemes, and we evaluate whether segments discovered by our approach correspond to surgemes. In Table 1, we compare the number of TSC segments for needle passing and suturing to the number of annotated surgeme segments. We apply different variants of the TSC algorithm and evaluate its ability to recover segments similar to surgemes. We consider: (Kin+Video), which is the full TSC algorithm; (Kin), which only uses kinematics; (Video), which only uses the visual annotations; (VELS), which uses the zero-crossing velocity heuristic to obtain the initial transitions; and (NO-H), which treats all of the variables as one big feature space and does not hierarchically cluster. A key difference between our segmentation and number of annotated surgemes is our compaction and pruning steps. To account for this, we first select a set of surgemes that are expressed in most demonstrations (i.e. simulating pruning), and we also apply a compaction step to the surgeme segments. When surgemes appear consecutively, we only keep the one instance of each. We explore two metrics: **seg-surgeme**, the fraction of TSC clusters with only one surgeme switch (averaged over all demonstrations); and **surgeme-seg**, the fraction of surgeme switches that fall inside exactly one TSC cluster.

We found that the transitions learned by TSC with both the kinematic and video features were the most aligned with the surgemes. Here 83% and 73% of transition clusters for needle passing and suturing respectively contained exactly one surgeme transition when both were used. For the needle passing task, we found that the video features alone could give a reasonably accurate segmentation. However, this did not hold for the suturing dataset. The manual video features are low dimensional and tend to under-segment. For the suturing dataset, a combination of the visual and kinematic features was most aligned with the surgemes. Similarly, this scaling problem affects the variant that does not hierarchically cluster, leading to a small number of clusters, and inaccuracy.

9. Future work

These results suggest several avenues for future work. First, we will explore using convolutional neural networks to automatically extract visual features for segmentation. This will alleviate a key challenge in applying TSC to new datasets. We will also explore how other results in deep learning such as autoencoders and recurrent networks can be used to segment data without linearity assumptions. Segmentation is the first step in a broader robot learning pipeline, and we are actively exploring using segmentation to construct rewards for reinforcement learning.

10. Conclusion

We have presented TSC, which leverages the consistent structure of repeated demonstrations to robustly learn segmentation criteria. To learn these clusters, TSC uses a hierarchical DP-GMM with a series of merging and pruning steps. Our results on a synthetic example suggest that this approach is more robust than five other segmentation algorithms. We further applied our algorithm to three surgical datasets and found that the transition state clusters correspond well to manual annotations and transitions with respect to motions from a pre-defined surgical motion dictionary (surgemes).

Acknowledgments

We are grateful to Florian Pokorny, Jeff Mahler, and Michael Laskey for feedback and discussions.

Funding

The author(s) disclosed receipt of the following financial support for the research, authorship, and/or publication of this article: This research was supported in part by a seed grant from the UC Berkeley Center for Information Technology in the Interest of Society (CITRIS), the UC Berkeley AMPLab, and by the US National Science Foundation (Award IIS-1227536: Multilateral Manipulation by Human-Robot Collaborative Systems). This work has been supported in part by funding from Google and Cisco.

References

Alvarez M, Peters JR, Lawrence ND and Schölkopf B (2010) Switched latent force models for movement segmentation. In: *Advances in Neural Information Processing Systems*, pp. 55–63.

Alvarez MA, Luengo D and Lawrence ND (2013) Linear latent force models using gaussian processes. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35(11): 2693–2705.

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.

Barbič J, Safonova A, Pan JY, Faloutsos C, Hodgins JK and Pollard NS (2004) Segmenting motion capture data into distinct behaviors. In: *Proceedings of Graphics Interface 2004*. Canadian Human–Computer Communications Society, pp. 185–194.

Brooks R (1986) A robust layered control system for a mobile robot. *IEEE Journal on Robotics and Automation* 2(1): 14–23.

Calinon S and Billard A (2004) Stochastic gesture production and recognition model for a humanoid robot. In: *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Sendai, Japan, 28 September–2 October 2004, pp. 2769–2774.

Calinon S, D'halluin F, Sauser EL, Caldwell DG and Billard AG (2010) Learning and reproduction of gestures by imitation. *IEEE Robotics and Automation Magazine* 17(2): 44–54.

Chiappa S and Peters JR (2010) Movement extraction by detecting dynamics switches and repetitions. In: *Advances in Neural Information Processing Systems*. pp. 388–396.

Chuck C, Laskey M, Krishnan S, Joshi R, Fox R and Goldberg K (2017) Statistical data cleaning for deep learning of automation tasks from demonstrations. In: *Conference on Automation Sciences and Engineering CASE 2017*.

Faria DR, Martins R, Lobo J and Dias J (2012) Extracting data from human manipulation of objects towards improving autonomous robotic grasping. *Robotics and Autonomous Systems* 60(3): 396–410.

Fikes RE, Hart PE and Nilsson NJ (1972) Learning and executing generalized robot plans. *Artificial intelligence* 3: 251–288.

Fod A, Matarić MJ and Jenkins OC (2002) Automated derivation of primitives for movement classification. *Autonomous robots* 12(1): 39–54.

Fox E, Jordan MI, Sudderth EB and Willsky AS (2009) Sharing features among dynamical systems with beta processes. In: *Advances in Neural Information Processing Systems*, pp. 549–557.

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

Gehrig D, Krauthausen P, Rybok L, et al. (2011) Combined intention, activity, and motion recognition for a humanoid household robot. In: *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, pp. 4819–4825.

Ghahramani Z and Jordan MI (1993) Supervised learning from incomplete data via an EM approach. In: *Advances in Neural Information Processing Systems 6, [7th NIPS Conference]*, Denver, CO, 1993, pp. 120–127.

Ijspeert A, Nakanishi J and Schaal S (2002) Learning attractor landscapes for learning motor primitives. In: *Neural Information Processing Systems (NIPS)*, pp. 1523–1530.

Konidaris G, Kuindersma S, Grupen R and Barto A (2011) Robot learning from demonstration by constructing skill trees. *The International Journal of Robotics Research* 31(3): 360–375.

Krishnan S, Garg A, Patil S, et al. (2015) Transition state clustering: Unsupervised surgical trajectory segmentation for robot learning. In: *International Symposium of Robotics Research*. Springer STAR.

Kruger V, Herzog D, Baby S, Ude A and Kragic D (2010) Learning actions from observations. *IEEE Robotics and Automation Magazine* 17(2): 30–43.

Krüger V, Tikhanoff V, Natale L and Sandini G (2012) Imitation learning of non-linear point-to-point robot motions using Dirichlet processes. In: *2012 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, pp. 2029–2034.

Kulic D, Takano W and Nakamura Y (2009) Online segmentation and clustering from continuous observation of whole body motions. *IEEE Transactions on Robotics* 25(5): 1158–1166.

Kulis B and Jordan MI (2012) Revisiting *k*-means: New algorithms via Bayesian nonparametrics. In: *Proceedings of the 29th International Conference on Machine Learning (ICML 2012)*, Edinburgh, Scotland, UK, 26 June–1 July 2012.

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: *Proceedings of WACV*.

Lee SH, Suh IH, Calinon S and Johansson R (2015) Autonomous framework for segmenting robot trajectories of manipulation task. *Autonomous Robots* 38(2): 107–141.

Liang J, Mahler J, Laskey M, Li P and Goldberg K (2017) Using dVRK teleoperation to facilitate deep learning of automation tasks for an industrial robot. Research report, UC Berkeley. Available at: http://goldberg.berkeley.edu/pubs/2017-Liang-DY-Teleop_CASE_CAMERA_READY.pdf

Lin HC, Shafran I, Murphy TE, Okamura AM, Yuh DD and Hager GD (2005) Automatic detection and segmentation of robot-assisted surgical motions. In: *Proceedings 8th International Conference Medical Image Computing and Computer-Assisted Intervention (MICCAI 2005)*, Palm Springs, CA, 26–29 October 2005, Part I, pp. 802–810.

Lin HC, Shafran I, Yuh D and Hager GD (2006) Towards automatic skill evaluation: Detection and segmentation of robot-assisted surgical motions. *Computer Aided Surgery* 11(5): 220–230.

Lin JFS, Joukov V and Kulic D (2014) Full-body multi-primitive segmentation using classifiers. In: *2014 14th IEEE-RAS International Conference on Humanoid Robots (Humanoids)*. IEEE, pp. 874–880.

Lin JFS, Karg M and Kulić D (2016) Movement primitive segmentation for human motion modeling: A framework for analysis. *IEEE Transactions on Human–Machine Systems* 46(3): 325–339.

Lioutikov R, Neumann G, Maeda G and Peters J (2015) Probabilistic segmentation applied to an assembly task. In: *2015 IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids)*. IEEE, pp. 533–540.

Mao R, Yang Y, Fermüller C, Aloimonos Y and Baras JS (2014) Learning hand movements from markerless demonstrations for humanoid tasks. In: *2014 14th IEEE-RAS International Conference on Humanoid Robots (Humanoids)*. IEEE, pp. 938–943.

Meier F, Theodorou E and Schaal S (2012) Movement segmentation and recognition for imitation learning. In: *Artificial Intelligence and Statistics*, pp. 761–769.

Meier F, Theodorou E, Stulp F and Schaal S (2011) Movement segmentation using a primitive library. In: *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, pp. 3407–3412.

Moeslund TB and Granum E (2001) A survey of computer vision-based human motion capture. *Computer Vision and Image Understanding* 81(3): 231–268.

Morasso P (1983) Three dimensional arm trajectories. *Biological Cybernetics* 48(3): 187–194.

Murali A, Sen S, Kehoe B, et al. (2015) Learning by observation for surgical subtasks: Multilateral cutting of 3D viscoelastic and 2D orthotropic tissue phantoms. In: *IEEE International Conference on Robotics and Automation (ICRA 2015)*, Seattle, WA, 26–30 May 2015, pp. 1202–1209.

Niekum S and Chitta S (2013) Incremental semantically grounded learning from demonstration. In: *Proceedings of RSS 2013*.

Niekum S, Osentoski S, Konidaris G and Barto AG (2012) Learning and generalization of complex tasks from unstructured demonstrations. In: *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2012)*, Vilamoura, Algarve, Portugal, 7–12 October 2012, pp. 5239–5246.

Paraschos A, Daniel C, Peters J and Neumann G (2013) Probabilistic movement primitives. In: Burges C, Bottou L, Welling M, Ghahramani Z and Weinberger K (eds.) *Advances in Neural Information Processing Systems 26*, pp. 2616–2624.

Pastor P, Kalakrishnan M, Meier F, et al. (2013) From dynamic movement primitives to associative skill memories. *Robotics and Autonomous Systems* 61(4): 351–361.

Quellec G, Lamard M, Cochener B and Cazuguel G (2014) Real-time segmentation and recognition of surgical tasks in cataract surgery videos. *IEEE Transactions on Medical Imaging* 33(12): 2352–2360.

Saeedi A, Hoffman M, Johnson M and Adams R (2016) The segmented ihmm: A simple, efficient hierarchical infinite HMM. *arXiv preprint arXiv:1602.06349*.

Sahbani A, Dias J and Menezes P (2008) Grasp and task learning by imitation. In: *IROS-2008 Workshop on Grasp and Task Learning By Imitation*.

Schaal S (2006) Dynamic movement primitives-a framework for motor control in humans and humanoid robotics. In: *Adaptive Motion of Animals and Machines*. Springer, pp. 261–280.

Sternad D and Schaal S (1999) Segmentation of endpoint trajectories does not imply segmented control. *Experimental Brain Research* 124(1): 118–136.

Sung C, Feldman D and Rus D (2012) Trajectory clustering for motion prediction. In: *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, pp. 1547–1552.

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.

Tao L, Zappella L, Hager GD and Vidal R (2013) Surgical gesture segmentation and recognition. In: *Medical Image Computing and Computer-Assisted Intervention (MICCAI 2013)*. Springer, pp. 339–346.

Ude A, Gams A, Asfour T and Morimoto J (2010) Task-specific generalization of discrete and periodic dynamic movement primitives. *IEEE Transactions on Robotics* 26(5): 800–815.

Ude A, Nemec B, Petrić T and Morimoto J (2014) Orientation in cartesian space dynamic movement primitives. In: *2014 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, pp. 2997–3004.

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* 42(4): 1039–1052.

Varadarajan B, Reiley C, Lin H, Khudanpur S and Hager G (2009) Data-derived models for segmentation with application to surgical assessment and training. In: *Medical Image Computing and Computer-Assisted Intervention (MICCAI 2009)*. Springer, pp. 426–434.

Viviani P and Cenzato M (1985) Segmentation and coupling in complex movements. *Journal of Experimental Psychology: Human Perception and Performance* 11(6): 828.

Volkov M, Rosman G, Feldman D, Fisher JW and Rus D (2015) Coresets for visual summarization with applications to loop closure. In: *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, pp. 3638–3645.

Wächter M and Asfour T (2015) Hierarchical segmentation of manipulation actions based on object relations and motion characteristics. In: *2015 International Conference on Advanced Robotics (ICAR)*. IEEE, pp. 549–556.

Willsky AS, Sudderth EB, Jordan MI and Fox EB (2009) Sharing features among dynamical systems with beta processes. In: *Advances in Neural Information Processing Systems*, pp. 549–557.

Wu C, Zhang J, Savarese S and Saxena A (2015) Watch-n-patch: Unsupervised understanding of actions and relations. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4362–4370.

Zappella L, Haro BB, Hager GD and Vidal R (2013) Surgical gesture classification from video and kinematic data. *Medical Image Analysis* 17(7): 732–745.

Appendix A: An introduction to variational inference

Let $\mathcal{X}$ be a set of random variables $\{X_1, X_2, \ldots, X_N\}$. There is a probability density $p(\cdot)$ defined over the domain of $\mathcal{X}$:

$$p(\mathcal{X}) = p(x_1, x_2, \ldots, x_N)$$

Suppose that only a subset of the random variables are observed $\mathcal{O} \subset \mathcal{X}$.

In Bayesian parameter inference, we are interested in the posterior distribution of the remaining unobserved random variables $\mathcal{H} = \mathcal{X} - \mathcal{O}$:

$$p(\mathcal{H} \mid \mathcal{O}) = \frac{p(\mathcal{X})}{\int_{\mathcal{H}} p(\mathcal{X})\,\partial\mathcal{H}} = \frac{p(\mathcal{X})}{p(\mathcal{H})}$$

This expression is usually intractable due to difficulty of integrating over the entire parameter space in the denominator $p(\mathcal{H})$. One approach is to use MCMC to approximately compute the integral in the denominator by sampling from the distribution $p(\mathcal{H})$. This approach, while widely used, can be very slow and inefficient.

In variational inference, instead of sampling to calculate $p(\mathcal{H})$, a parametrized version of the distribution is optimized. Let us instead pretend that we had choice over $q(\cdot) \approx p(\cdot)$, a density over the hidden variables $\mathcal{H}$:

$$p(\mathcal{H}) = \int_{\mathcal{H}} p(\mathcal{X})\,\partial\mathcal{H}$$

$$p(\mathcal{H}) = \int_{\mathcal{H}} p(\mathcal{X})\,\frac{q(\mathcal{H})}{q(\mathcal{H})}\,\partial\mathcal{H}$$

$$p(\mathcal{H}) = \mathbf{E}_q\left(\frac{p(\mathcal{X})}{q(\mathcal{H})}\right)$$

$$\log p(\mathcal{H}) = \log \mathbf{E}_q\left(\frac{p(\mathcal{X})}{q(\mathcal{H})}\right)$$

This gives a lower bound via Jensen's inequality:

$$\log p(\mathcal{H}) \geq \mathbf{E}_q(\log p(\mathcal{X})) - \mathbf{E}_q(\log q(\mathcal{H}))$$

This function is called the evidence lower bound (ELBO) function:

$$L(q) = \mathbf{E}_q(\log p(\mathcal{X})) - \mathbf{E}_q(\log q(\mathcal{H}))$$

The key idea in variational inference is to choose a $q(\mathcal{H})$ that approximates $p(\mathcal{H})$ but is tractable to compute the expectations $\mathbf{E}_q$. This function has two terms: (1) the likelihood term evaluating whether the distribution is supported by p; (2) the entropy term maximizing the entropy of hidden variables.

A.1. Mean-field approximation

One way to select $q(\cdot)$ is to use a density where all the random variables are conditionally independent:

$$q(\mathcal{H}) = \Pi_{X_i \in \mathcal{H}} q_i(X_i)$$

Then, it follows that the "entropy" term of the ELBO function is

$$\sum_{i \in \mathcal{H}} \mathbf{E}_i \log q_i(X_i)$$

For the "likelihood" term, we can apply the chain rule to $p(\mathcal{X})$, to obtain

$$p(\mathcal{X}) = p(\mathcal{H} \mid \mathcal{O})\, p(\mathcal{O})$$

$$p(\mathcal{X}) = p(\mathcal{O})\, \Pi_i p(\mathcal{H}_i \mid \mathcal{H}_{j \neq i}, \mathcal{O})$$

It follows that the mean-field ELBO function is

$$L(q) = \log(p(\mathcal{O})) + \sum_{i \in \mathcal{H}} \mathbf{E}_i \log p(\mathcal{H}_i \mid \mathcal{H}_{j \neq i}, \mathcal{O}) + \mathbf{E}_i \log q_i(X_i)$$

which becomes

$$L(q) = const + \sum_{i \in \mathcal{H}} \mathbf{E}_i \log p(\mathcal{H}_i \mid \mathcal{H}_{j \neq i}, \mathcal{O}) + \mathbf{E}_i \log q_i(X_i)$$

A.2. Variational inference

The goal of variational inference is to optimize the ELBO function:

$$\max_q L(q)$$

In principle, we could apply standard techniques such as gradient descent, suppose we had q parametrized by some λ:

$$\lambda^{(k)} = \lambda^{(k-1)} - \lambda \nabla_{\lambda} L(q_{\lambda})$$

However, since we are really optimizing over the space of distributions, this has some complicated issues. In other words, q is not a Euclidean point, it is a distribution, and accordingly this changes the metric space. The gradient is weighted accordingly with a matrix $G(\lambda)^{-}1$. This is analogous to gradient descent on a manifold where distances are measured by different metric. It turns out that if you use the Kullback–Leibler (KL) divergence as this metric, $G(\lambda)$ is a matrix called the Fisher information matrix. Then, the updates have a really straightforward form.

Appendix B: Mean-field variational inference For DP-GMM

To derive a mean-field variational procedure for the DP-GMM, we first define a generative model:

Hyper-parameters: α initial concentration parameter, G_0 base distribution of Gaussian parameters, K max number of clusters.

1. Draw $V_i \sim Beta(1, \alpha), i = 1, 2, \ldots, K$
2. $\theta = V_i \prod_{j=1}^{i-1}(1 - V_i)$
3. Draw $\eta_i \sim G_0, i = 1, 2, \ldots, K$
4. For each data point n:
   (a) $Z_n \sim \textbf{Multinomial}(\theta)$
   (b) $X_n \sim \textbf{Normal}(\eta_{z_n})$

Optimization algorithm: Based on this generative model, we can derive the ELBO function:

$$\begin{aligned}\log p(\mathbf{X}|\alpha, G_0) \geq\ & E[\log p(\mathbf{V}|\alpha)] \\ & + E[\log p(\boldsymbol{\eta}|G_0)] \\ & + \sum_{i=0}^{N} E[\log p(Z_n|\mathbf{V})] \\ & + E[\log p(x_n|Z_n)] \\ & - E[\log q(\mathbf{Z}, \mathbf{V}, \boldsymbol{\eta})]\end{aligned}$$

Then, we approximate q with a mean-field approximation:

$$q(\mathbf{Z}, \mathbf{V}, \boldsymbol{\eta}) = \prod_{i=1}^{K} q_1(v_i) \prod_{i=1}^{K} q_2(\eta_i) \prod_{i=1}^{N} q_3(z_n)$$

To infer the parameters of this generative model, we apply the following coordinate ascent algorithm:

$$q(zn = i) = \phi_{n,i}$$

$$q(zn > i) = \sum_{i=1}^{K} \phi_{n,i}$$

$$E[\log V_i] = \textbf{DiGamma}(\gamma_{i,1}) - \textbf{DiGamma}(\gamma_{i,1} + \gamma_{i,2})$$

$$E[\log(1 - V_i)] = \textbf{DiGamma}(\gamma_{i,2}) - \textbf{DiGamma}(\gamma_{i,1} + \gamma_{i,2})$$

$$\gamma_{i,1} = \sum_{n=0}^{N} \phi_{n,i}$$

$$\gamma_{i,2} = \alpha + \sum_{n=0}^{N} \sum_{j=i+1}^{K} \phi_{n,j}$$