Unsupervised Surgical Task Segmentation with Milestone Learning

Sanjay Krishnan*[1], Animesh Garg*[1,2], Sachin Patil[1], Colin Lea[3], Gregory Hager[3], Pieter Abbeel[1], Ken Goldberg[1,2]
* denotes equal contribution

Abstract Surgical tasks are complex multi-step sequences of smaller subtasks (often called surgemes) and it is useful to segment task demonstrations into meaningful subsequences for:(a) extracting finite-state machines for automation, (b) surgical training and skill assessment, and (c) task classification. Existing supervised methods for task segmentation use segment labels from a dictionary of motions to build classifiers. However, as the datasets become voluminous, the labeling becomes arduous and further, this method doesnt́ generalize to new tasks that dont́ use the same dictionary.
We propose an unsupervised semantic task segmentation framework by learning "milestones", ellipsoidal regions of the position and feature states at which a task transitions between motion regimes modeled as locally linear. Milestone learning uses a hierarchy of Dirichlet Process Mixture Models, learned through Expectation-Maximization, to cluster the transition points and optimize the number of clusters. It leverages transition information from kinematic state as well as environment state such as visual features. We also introduce a compaction step which removes repetitive segments that correspond to a mid-demonstration failure recovery by retrying an action. We evaluate Milestones Learning on three surgical subtasks: pattern cutting, suturing, and needle passing. Initial results suggest that our milestones qualitatively match manually annotated segmentation. While one-to-one correspondence of milestones with annotated data is not meaningful, the milestones recovered from our method have exactly one annotated surgeme transition in 74% (needle passing) and 66% (suturing) of total milestones, indicating a semantic match.

1 Introduction

There is a growing corpus of both trajectory and sensor data from robot-assisted minimally invasive procedures (RMIS). Numerous data-driven methodologies for RMIS have been proposed in recent research including: extraction of finite-state machines for automation [9, 17, 18], surgical skill assessment [16], and task classification [27, 30]. Surgical *demonstrations*, repeated executions of a particular task in a consistent environment by a human teleoperator, are a useful type of RMIS data. However, even in a consistent environment, training models on raw trajectory and sensor data is challeng-

[1] EECS, [2]IEOR, UC Berkeley, e-mail: {sanjaykrishnan,animesh.garg,sachinpatil,pabbeel,goldberg}@berkeley.edu
[3] Computer Science Department, The Johns Hopkins University, e-mail: clea1@jhu.edu, hager@cs.jhu.edu

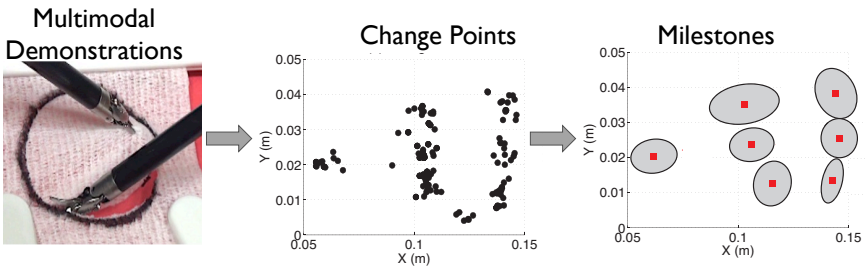

Fig. 1: Primary steps in milestone learning: (1) identify change points in multimodal demonstrations by using kinematic and environment features, and (2) cluster them into milestones. The result is a set of clusters in state-space. Milestones signify transitions in phases across the examples.

ing as most surgical tasks are complex, multi-step procedures. An important first step is segmentation, which allows for local modeling of subtasks. While segmentation of surgical data is a well studied problem [16, 24, 30], existing methods are *supervised*. Supervised segmentation relies on either manually annotated example segments, or a dictionary of pre-identified surgical motions ("surgemes"). Acquiring a sufficient amount of labeled training data to span all possible events in the demonstrations can be time-consuming.

In this paper, we explore applying an *unsupervised* methodology to segment surgical tasks into smaller phases. The goal is to learn the consistently recurring spatio-temporal transitions in a series of demonstrations *without* a motion dictionary or labeled segments. Unsupervised segmentation allows for more robustness to changes in motion or environment without having to provide more labeled training data. This is particularly useful for building finite state machines for automation, which currently requires time-consuming manual analysis of surgical videos and trajectory data[18].

Unsupervised segmentation methods rely on an assumed structure of the data, typically local linearity as in autoregression [20, 28], to group together time points that share that same structure. Likewise, in this work, we consider a locally linear model. However, the RMIS setting presents new challenges not considered in prior work. First, the relevant features for surgical segmentation are multimodal. Recent results have established the benefit of using both kinematic data and video in supervised segmentation [12, 25]. Second, surgical tasks are often long, multi-step procedures with a large temporal variation. One potentially challenging feature of surgical demonstration data is retrial-until-success operations. For example, a human tele-operator may continuously try to pass a needle through a loop and repeat until she succeeds. There might be a varying number of retrials in different demonstrations, and we would like to identify and eliminate the failures.

To address these challenges, we present *milestone learning*. Each milestone is a region of the state-space (including both robot pose and visual features) where demonstrations transition between locally linear motion regimes. We use two visual features in all three tasks: object grasp events and surface penetration. The additional features allow for greater discrimination of robot actions and effects on the environment, and we evaluate the results with and without the features in figure 3. Since the primary focus of this work is evaluation of milestone learning framework we use visual features annotations, and defer autonomous surgical perception to future work. However it is

worth noting, that both of our visual features can be automatically constructed based on recent computer vision results [15].

As illustrated in in Figure 1, milestones are learned with a hierarchical model:
1. **Change Points:** Identify locally linear transition regimes and mark *change points*.
2. **Compaction:** Remove segments that correspond to failure and retrial phases.
3. **State Clustering:** The remaining change points (after compactions) are clustered with respect to the robot's state at the change point.
4. **Time Clustering:** Within each state cluster, change points that happen at similar times are clustered. The highest level of clustering results in the *milestones*.

The hierarchy differentiates milestones from segmentations described in existing literature, as milestones are clusters of segment endpoints. Milestones define important transition regions for a task and not just a single trajectory. Milestones address the time variation problem in two ways: the compaction step ensures that between milestones there are the same number of actions, and the temporal clustering allows for continuous noise (i.e., slightly faster or slower). They are robust to small amounts of temporal distortion both discrete and continuous. However, this hierarchical model is not without its own challenges. In a typical clustering formulation like K-Means or a Gaussian Mixture Model, we would have to select the number of clusters. To address this concern, we leverage a non-parametric model called a Dirichlet Process, that places a prior probability on the number of clusters that adjusts with incoming data.

Contributions:
1. An unsupervised semantic task segmentation technique called "milestone learning" that uses a hierarchy of Dirichlet Process Mixture Models to identify spatio-temporal regions where demonstrations transition between locally linear regimes. To capture robot-environment interactions, we leverage kinematic data $\big(x(t)\big)$ with a set of environment features $\big(z(t)\big)$ obtained by fusing kinematic and visual information.
2. A compaction procedure which removes repetitive segments that correspond to a mid-demonstration failure recovery by multiple retrials of an action.
3. We evaluate our method on three surgical subtasks: pattern cutting, suturing, and needle passing; and compare with manually annotated segmentation.

2 Related Work

Segmentation in Vision and Machine Learning: Hierarchical segmentation models have been widely applied in computer vision [6, 11]. Each level of the hierarchy coarsens clustering, leading to different segmentation abstractions. As in our work, unsupervised approaches rely on identifying locally linear structures as in [4], where videos are modeled as transitions on a lower-dimensional linear subspace and segments are defined as changes in these subspaces.

Unsupervised segmentation models have also been well studied in the Machine Learning community. Willsky et al [28] proposed the Beta-Process Autoregressive Hidden Markov Model (BP-AR-HMM). This model fits an autoregressive model to time-series, where time $t+1$ is a linear function of times $t-k,\ldots,t$. The linear function transitions according to an HMM, and to avoid the extreme quantity of hyperparameters, they use a Beta-Process prior on the transitions. BP-AR-HMM model also allows for multiple active linear regimes at any instant of time. While, our method

models the linear function in terms of only one previous time step, and only one linear regime is active at a time. It is worth noting that the increased expressiveness of the BP-AR-HMM model comes at a cost of additional training data. Our experimental results suggest that our model is better conditioned on smaller datasets.

Segmentation in robotics: Robotics often poses challenges not addressed in vision. For example, many vision approaches are not robust to multiple moving objects [7]. One class of segmentation problems that has been extensively studied is Switched Linear Dynamical Systems (SLDS) [1, 14, 29]. Additionally, studies explore using HMMs to consider sequeneces of segments [2, 3, 12, 26]. However, most of the HMM work has focused on supervised segmentation, where segments are learned from a small set of labeled examples.

To apply HMM-style models to the unsupervised setting, we require hierarchies features, as HMMs model both global and local structure. Hierarchical models are a key part of unsupervised segmentation, as they allow for aggregations at different levels of abstraction. Konidaris et al. [10] proposed a technique called skill trees for reinforcement learning. While exploring a different problem, Konidaris et al. apply a hierarchical clustering methodology to learn the trees. Niekum et al. [21] apply a Bayesian non-parametric model (BP-AR-HMM) to segment demonstrations (a time-series of robot poses). After segmentation, they normalize each segment in a relative pose space w.r.t to each object in the environment. When these objects change, they can stitch the relative poses back together. Like the BP-AR-HMM model, this line work relies on a switched autoregression called, Piecewise ARX, where segments transition between different autoregression regimes. In this paper, we argue that surgical robotics requires additional environment specific features to capture the manipulation tasks and we evaluate this methodology against [21].

Motion Primitives and Learning From Demonstrations: Discrete step representation of tasks at different levels of hierarchy is key for interaction and learning both on part of teacher and learner. In robotics, learning such task representations are essential for capturing instructive demonstrations and generalization over multiple demonstrations [19]. A relevant line of work is motion primitive-based path planning and learning from demonstration (LfD) frameworks. Motion primitives are higher level actions, that are used to discretize the action space. Using motion primitives in LfD was initially proposed by Ijspeert et al. [8]. They argued that while LfD was a theoretically attractive model, albeit in practice, it suffered inefficiency in high-dimensional action spaces. They proposed a model where parametrized the action space in terms of motor primitives and learned a policy over the parameters rather than the high-dimensional action space. This work was extended by Pastor et al. [22], who proposed Dynamic Motion Primitives (DMP) which stitch together motion primitives into policies.

Surgical Task Recognition: In Surgical Robotics, researchers have studied the related problem of classifying and evaluating tele-operated surgery. This is often done with a domain specific high-level motion primitives for surgery called "surgemes". In Lin et al. [16, 27], the authors take a 78-dimensional state space of da Vinci manipulator motions and use Linear Discriminant Analysis (LDA) to project this state space into a 6-dimensional space in which they can separate clusters of surgical motions and associate the motions with surgemes. Zappella et al. [30] explored this problem using both Kinematic and Video data. Given manually segmented videos, they use features from both the videos and kinematic data to classify surgical motion. Quellec et al. [23] pro-

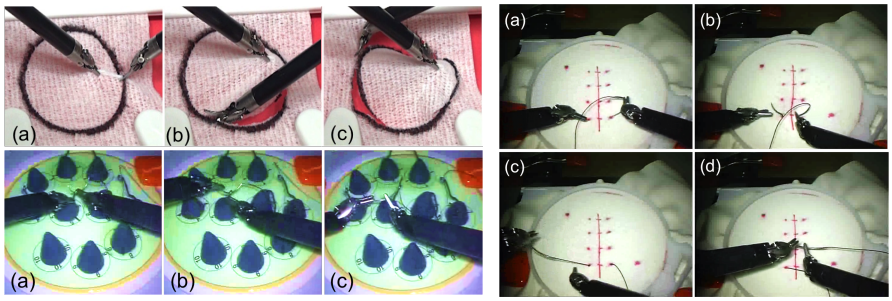

Fig. 2: **1. Circle Cutting:** (a) Notching to begin a cut, (b) Cutting clockwise, and (c) Counter clockwise cut to finish.
2. Needle Passing: (a) Pass needle through a loop (b) pull needle through (c) hand off the needle.
3. Suturing: (a) Penetrate one of the points, (b) Pass needle through the tissue phantom, (c) pull needle out of the phantom, (d) hand off the needle.

pose segmentation and recognition of surgical tasks in cataract surgery videos. Their method reuses previously archived videos to automatically analyze the current surgery, by analogy reasoning. Tao et al. [25] also propose a segmentation and recognition of surgical tasks using semi-markov conditional random fields on surgical kinematic and video data. This entire line of work requires some level of supervision either through a dictionary of surgemes or annotated example demonstrations.

3 Problem and Background

In this work, we evaluate our segmentation methodology on three tasks: circle cutting, needle passing, and suturing (Figure 2). To introduce the concepts and definitions in the work, we will use the circle cutting as a running example. In this task, a circle is marked on a sheet of gauze. The robot has one arm that is fitted with a cutting tool, and it has to cut along the circle. It first grips the gauze with the non-cutting arm (left), then uses the cutting tool to penetrate the gauze and cut a notch. Using this notch, it cuts along the circle.

3.1 State Space

Robot: We denote the robot's configuration at every time t with $x(t)$. In this work, this configuration is the pose of the end-effector which is a vector of the 3D translational position and 3D rotational position. For multilateral tasks, we apply our methodology independently on each arm.

Manipulation Features: At every time t, there is a feature vector $z(t)$ which represents observations of the environment by the sensors. We use the same features for all three of our tasks:

1. *Gripper grasp*. 1 if there is an object between the left gripper, 0 if not.
2. *Plane Penetration*. If the robot has penetrated work surface, then the feature value is the depth of penetration with respect to the plane. Otherwise, the value is 0.

For initial experiments, we manually identified the features in associated frames. These features can be automatically constructed, for example in Lea et al. [15], the authors track a surgical needle in a bounding box which is sufficient to determining whether it

is grasped by the gripper. In this work, we do not want to conflate the perception problem and the unsupervised segmentation problem so we defer automatic construction of the features to future work.

To make $z(t)$ more concrete, let us consider our running example circle cutting task. During notching, *Gripper grasp* is 1, and the *plane penetration* feature is the penetration depth if the vision system detects the gauze is successfully notched. During cutting the *Gripper grasp* feature is 1, and the *plane penetration* feature is the vertical displacement from the plane. For the three tasks that we explore experimentally (circle cutting, needle passing, and suturing), these features are sufficient for segmentation.

In the rest of the paper, we will refer to the "state" as the augmented state of both the robot configuration and the manipulation features denoted by boldface $\mathbf{x}$:

$$\mathbf{x}(t) = \begin{pmatrix} x(t) \\ z(t) \end{pmatrix}$$

3.2 Milestones

In this work, we address the problem of task segmentation using an abstraction called milestones. We are given a set of demonstrations $\mathscr{D} = \{d_i\}$. Each demonstration is a time-series of states $\mathbf{x}(t)$. We model demonstrations as locally linear, that is state transitions linearly between time steps, and the transition can be time dependent with a finite set of transition functions shared by all demonstrations. There is also an i.i.d zero-mean additive Gaussian noise process $W(t)$ which accounts for both model-error and observation error.

$$\mathbf{x}(t+1) = A_i\mathbf{x}(t) + W(t) : A_i \in \{A_1, ..., A_k\}$$

Milestones are clusters of in state-space and in time that signify transitions between locally linear regimes in the time series $\mathbf{x}(t)$. Intuitively, either the robot is moving differently than before or it is manipulating its environment differently than before. Milestones are thus a tuple of an ellipsoidal (follows from the Gaussian model) region of state-space and a time interval.

Problem 1 (Milestone Clustering). Given a set of demonstrations $\mathscr{D}$, where each demonstration d_i is a time-series of states $\mathbf{x}(t)$, find a set of milestones $\mathscr{M}$. Each $m_i \in \mathscr{M}$ has an ellipsoidal region of state-space where demonstrations transition and a interval of time at which the transition happens.

4 Milestone Clustering

Our algorithm has four basic steps, as described following:

1. We define a column vector of the current state and the next state $n(t) = \binom{\mathbf{x}(t+1)}{\mathbf{x}(t)}$. We cluster the vectors $n(t)$ and change points are times when $n(t)$ is in a different cluster than $n(t+1)$. (Section 4.2).
2. The set of change points define segments in each demonstration. We remove i.e., compact, consecutive segments that are overly similar. (Section 4.3)
3. W cluster the remaining change points in the state space $\mathbf{x}(t+1)$. (Section 4.4)
4. We cluster the change points in each cluster temporally. (Section 4.5)

4.1 Background: Bayesian Statistics

One challenge with clustering is hyper-parameter selection, such as the number of clusters. Recent results in Bayesian statistics can mitigate some of these problems. The basic recipe is to define a generative model, and then use Expectation Maximization to fit the parameters of the model to observed data. The generative model that we will use is called a mixture model, which defines a probability distribution that is a composite of multiple distributions.

One flexible class of mixture models are Gaussian Mixture Models (GMM), which are described generatively as follows. We first sample some c from a categorical distribution, one that takes on values from (1...K), with probabilities ϕ, where ϕ is a K dimensional simplex:

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

Then, given the event $\{c = i\}$, we specify a multivariate Gaussian distribution:

$$x_i \sim N(\mu_i, \Sigma_i)$$

The insight is that a stochastic process called the Dirichlet Process (DP) defines a distribution over discrete distributions, and thus instead we can draw samples of $cat(K, \phi)$ to find the most likely choice of K via EM. The result is the following model:

$$(K, \phi) \sim DP(H, \alpha)$$

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

$$X \sim N(\mu_i, \Sigma_i)$$

After fitting the model, every observed sample of $x \sim X$ will have a probability of being generated from a mixture component $P(x \mid c = i)$. Every observation x will have a most likely generating component. It is worth noting that each cluster defines an ellipsoidal region in the feature space of x, because of the Gaussian noise model $N(\mu_i, \Sigma_i)$.

We denote this entire clustering method in the remainder of this work as DP-GMM. We use the same model at multiple levels of the hierarchical clustering and we will describe the feature space at each level. We use a MATLAB software package to solve this problem using a variational EM algorithm [13].

4.2 Change Point Identification

The first step is to identify a set of change points for each demonstration in our demonstrations $\mathscr{D}$. In the previous section, we introduced the following demonstration model:

$$\mathbf{x}(t+1) = A_i \mathbf{x}(t) + W(t)$$

Where $W(t)$ is i.i.d Gaussian noise. To learn this model, we can apply DP-GMM to the following vectors: $n(t) = \binom{\mathbf{x}(t+1)}{\mathbf{x}(t)}$. Each cluster signifies a different "locally linear" regime. When $n(t)$ is in a different cluster than $n(t+1)$, we call this point a change point. For every demonstration there will be a set of times at which there is a transition between locally linear regimes.

4.3 Change Point Compaction

Once we have change points for each demonstration, the next step is to remove change points that correspond to retry-until-success actions, which are prevalent in surgical demonstrations. We model this behavior as consecutive linear regimes oscillating, i.e.,

transition from i to j and then j to i. To account for this, for each demonstration, we define a segment $\mathbf{s}^{(j)}[t]$ of states between each change point. If two consecutive segments are similar, we only keep the latest change point.

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), which uses dynamic programming to find a most-likely time alignment of the two segments. Let $\mathbf{s}^{(j+1)}[t^*]$ be a time aligned (w.r.t 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. δ is chosen empirically and a larger δ leads to a sparser distribution of change points, and smaller δ leads to more change points. However, since we are removing points from a time-series this requires us to adjust the time scale. Thus, from every following observation, we shift the time stamp back by the length of the pruned segments.

4.4 State-Space Clustering

After compaction, there are numerous change points at different locations in the state-space. At each change point t there is a state $\mathbf{x}(t)$. If we model the states at change points as drawn from a GMM model:

$$x(t) \sim N(\mu_i, \Sigma_i)$$

Then, we can apply the DP-GMM approach to cluster the state vectors at the change points. Each cluster defines an ellipsoidal region of the state-space space.

4.5 Time Clustering

Without both space and time, the transitions may be ambiguous. For example, in circle cutting, the robot may pass over a point twice in the same task. The milestone clustering problem clusters these change points together in both space and in time. The challenge is that we cannot naively use time as another feature, since it is unclear what metric to use to compare distance between $\binom{\mathbf{x}(t)}{t}$. However a second level of clustering by time within each state-space cluster can overcome this issue.Within a state cluster, if we model the times which change points occur as drawn from a GMM:

$$t \sim N(\mu_i, \sigma_i)$$

then we can apply DP-GMM to the set of scalar values of times $\{t\}$. This groups together events that happen at similar times during the demonstrations. The result is clusters of states and times. Thus, a milestone m_k is defined as tuple of an ellipsoidal region of the state-space and a time interval:

$$m_k = \left(ellipse(\mu_k, \Sigma_k), [t-\varepsilon, t+\varepsilon] \right) \quad \blacksquare$$

One advantage of the DP-GMM approach is that it allows us to cluster without specifying the number of clusters a priori. Even so, we may have many small clusters of outlier transition points. We found that in practice, we could prune the clusters by setting a rule that every cluster has to have a transition point from $\rho\%$ of the demonstrations (e.g 80% used in this work), and removed clusters that did not satisfy this

criterion. Like δ, ρ is another user-defined parameter for the algorithm which is set empirically. A larger ρ means less milestones and a smaller ρ means more milestones.

5 Applications of Milestones

Our framework is motivated by three potential applications of milestones as described:

1. Unsupervised Task Classification: The set of learned milestones divides a task into a small set of segments. We can apply this model to classify segments of a new demonstration, which is an inverse problem from the learning problem. The first step is to generate the transition features $n(t) = \binom{\mathbf{x}(t+1)}{\mathbf{x}(t)}$ for this new demonstration. Then, we assign each $n(t)$ to the mixture components learned in our milestone model to identify the change points.

The algorithm is iterative starting at the earliest change point. We assign each change point to a milestone if it falls in the spatial region and temporal interval. If a preceding change point assigned to a milestone and the current change point falls within the same milestone's spatial region (but not necessarily the temporal interval), we replace the assignment with the later change point. At the end there at most one change point assigned to a milestone, and thus defines a segmentation that follows from our hierarchical model.

2. Features For Skill Assessment: While the corpus of surgical data is growing, problems in this domain (such as skill assessment) often have very high-dimensional feature spaces and a relatively small number of demonstrations. Like other unsupervised learning techniques, milestones can also be used for low-dimensional feature learning. The event of reaching or missing a milestone in a task can be an important feature for classification, rather than using a very large feature space with many potentially irrelevant features. Milestones synthesize the most important transitions from all of the demonstrations.

3. Finite State Machines for Automation: Milestones are an abstraction that aid in the development of Finite State Machines. FSMs are widely applied in the automation of surgical subtasks [9, 17, 18]. The challenge is identifying important states for error-handling and recovery. Milestones naturally describe transitions where either the robot's state or interaction with the environment is transitioning. These are often the points where failures happen and the automation needs recovery logic.

6 Results and Discussion

We first evaluate the performance of our change point identification model by comparing against an existing segmentation model, BP-AR-HMM [21]. Next, we evaluate whether our milestone clusters correspond to manually annotated task structures. Then, we evaluate the sensitivity of the framework to compaction and pruning. Finally, we evaluate the match of the learned segments with the manually annotated surgemes in the JIGSAWS dataset [5].

6.1 Evaluation Tasks

Circle Cutting: In this task (Figure 2.1), we have a 5cm diameter circle drawn on a piece of gauze. The first step is to cut a notch into the circle (Figure 2.1a). The second

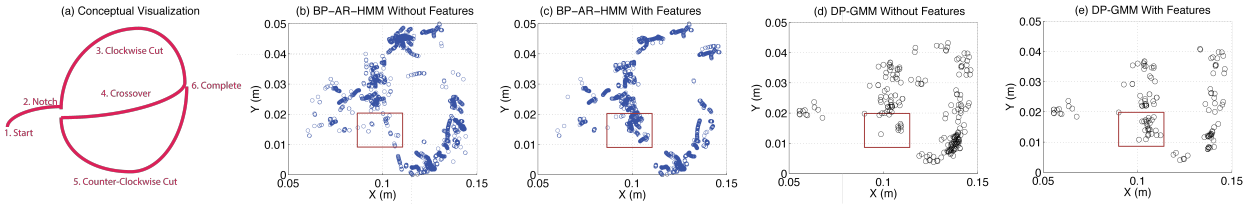

Fig. 3: In this figure, we plot the change points identified by the different methods. (a) First, we illustrate the circle cutting task conceptually. (b) We apply the BP-AR-HMM technique segmenting only on end-effector pose. (c) Next, we evaluate apply BP-AR-HMM to the multimodal features. In the red box, we mark a key transition point (transition from clockwise to counterclockwise cut) that is not detected without the features. (d-e) We run our technique with the DP-GMM with and without the features and find tighter clusters of change points.

step is to cut clockwise (Figure 2.1b). Next, the robot transitions to the other side cutting counter clockwise (Figure 2.1c). Finally, the robot finishes the cut at the meeting point of the two incisions. As the left arm's only action is maintain the gauze in tension, we exclude it from the analysis. For the circle cutting task, we collected 10 demonstrations by non-experts familiar with operating the da Vinci Research Kit (dVRK).

Needle Passing: We applied our framework to 40 demonstrations of the needle passing task. Figure 2.2 illustrates the three main steps of this task which are repeated 4 times for each of the loops. In Figure 2.2a, the robot passes a needle through a loop using its right arm, then its left arm to pull the needle through the loop as in Figure 2.2b. Finally, the robot hands the needle off from the left arm to the right arm (Figure 2.2c).

Suturing: Next, we explored a 4 throw suturing task. We illustrate the task in Figure 2.3. Using the right arm, the first step is to penetrate one of the points on right side (Figure 2.3a). The next step is to force the needle through the phantom to the other side (Figure 2.3b). Using the left arm, the robot pulls the needle out of the phantom (Figure 2.3c), and then hands it off to the right arm for the next point (Figure 2.3d).

6.2 Experiment 1. Segmentation Model Evaluation

We evaluate the change point model described in this work against a recently proposed algorithm (BP-AR-HMM [21]). There are a few main differences between BP-AR-HMM model and our model (which we call DP-GMM). First, BP-AR-HMM was only applied to the robot state-space $x(t)$, and in this work, we argue that augmenting the state with the additional features $z(t)$ improves change point detection. Next, our model compacts together similar change points based on DTW. Finally, while BP-AR-HMM is also based on a non-parameteric bayesian process but it has many more parameters to learn as it models multiple simultaneously active dynamical regimes.

We evaluate these differences on the circle cutting task as seen in Figure 3. In Figure 3a, we illustrate a conceptual illustration of the circle cutting task on a planar work surface. In Figure 3b, we first apply BP-AR-HMM to only the robot states. At each point where the model transitions, we mark the end-effector (x,y,z) location (note that the model is trained on poses in $SE(3)$). While the change points do recover some of the task structure they are not well clustered and there are many outliers. Next, in Figure 3c, we apply BP-AR-HMM to the multimodal states $z(t)$ in addition to $x(t)$. These additional states reduce the number of outliers dramatically. In particular, we

show a region (red box) to highlight the benefits of these features. When the circle task crosses over it has to re-enter the notch point and adjust to cut the other half of the circle. When only using the end-effector position, the locations where this transition happens is unreliable as operators may approach the entry from slightly different angles. On the other hand, the use of a gripper contact binary feature clusters the change points around the point at which the gripper is in position and ready to begin cutting again. Figure 3d, illustrates the benefits of our DP-GMM model over BP-AR-HMM even without the extra features. DP-GMM uses the combination of the Gaussian Mixture clustering and compaction, and gives fewer & more tightly clustered transitions. Figure 3e, finally shows the entire model we use in this paper, DP-GMM with the environment features. This model gives the tightest clustering & the fewest number of change points.

To give some intuition on why we get improved results, we illustrate the trajectories of the 10 cutting demonstrations in Figure 4. We find that the trajectories are quite noisy. The compaction step in conjunction with our DP-GMM clustering allows us to prune out oscillations between two dynamical regimes, i.e., the operator moves off the circle and then compensates by moving back. We evaluate the benefits of compaction in more detail in the subsequent experiments in section 6.5.

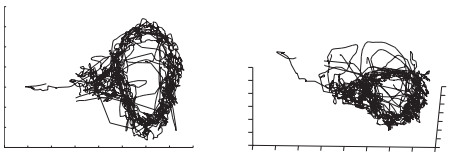

Fig. 4: We illustrate the variations in the circle cutting demonstrations visualized in two views (X-Y and X-Y-Z).These variations result in non-essential transitions in manually annotated surgemes. Our method, DP-GMM attempts to smooth out the noise with the use of compaction and pruning, and learns consistently recurring transitions as milestones.

6.3 Experiment 2. Milestone Evaluation

Next, we evaluate how these change points cluster into milestones.

Circle Cutting: We revisit the circle cutting change points from the previous experiment and evaluate how the change points cluster into milestones. In particular, we want to evaluate how well change points correspond to manually identified segments. In Figure 5a, we mark 6 manually identified transitions points for this task from [18]: (1) start, (2) notch, (3) finish 1st cut, (4) cross-over, (5) finish 2nd cut, and (6) connect the two cuts. Figure 5b shows the change points obtained from our algorithm. And Figure 5c shows the milestone clusters learned (numbered by time interval midpoint). The algorithm found 8 clusters, one of which was pruned out using our 80% threshold rule.

The remaining 7 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, the milestone framework separates them into different clusters even though a human annotator may not do so.

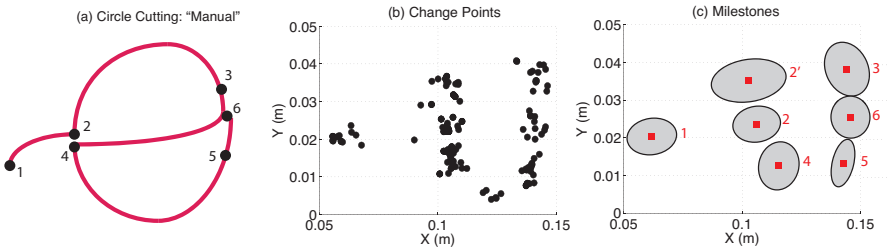

Fig. 5: (a) A manual semantic segmentation of circle cutting task. (b) The change points for the circle cutting task are marked in black. (c) The milestones, which are clusters of the change points, are illustrated with their 75% confidence ellipsoid.

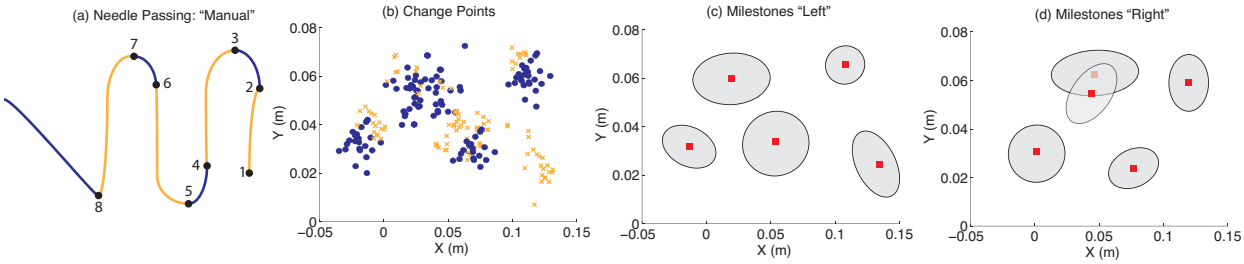

Fig. 6: (a) A manual semantic segmentation of needle passing task. In orange, we mark the left arm motions and in blue we mark the right arm motions (b) The change points for the task are marked in orange (left arm) and blue (right arm). (c-d) The milestones, which are clusters of the change points, are illustrated with their 75% confidence ellipsoid for both arms

Needle Passing: Next, we apply our methodology to the needle passing task. In Figure 6a, we illustrate a manual segmentation of the needle passing task. This task requires multilateral manipulation so we find milestones for both arms. We color code interactions of the right arm in blue and left arm in orange. In Figure 6b, we plot the change points in (x, y, z) end-effector space for both arms. We find that these change points correspond well to the logical segments of the task (Figure 6a). These demonstrations are noisier than the circle cutting demonstrations and there are more outliers. The subsequent clustering finds 9 milestones (2 pruned). Next, Figures 6c-d illustrate the milestone clusters. We find that again the milestones learn a small parametrization for the task structure with the milestones corresponding well to the manual segments. However, in this case, the noise does lead to a spurious milestone. One possible explanation is that the two middle loops are in close proximity and demonstrations contain many adjustments to avoid colliding with the loop and the other arm while passing the needle through leading to numerous change points in that location. Since we independently cluster for milestones on each arm, we are unable to learn that structure. A detailed study of this issue along with a multilateral milestone model is an interesting avenue of future work.

Suturing: In Figure 7, we show the change points and milestones for the suturing task. As before, we mark the left arm in orange and the right arm in blue. This task was

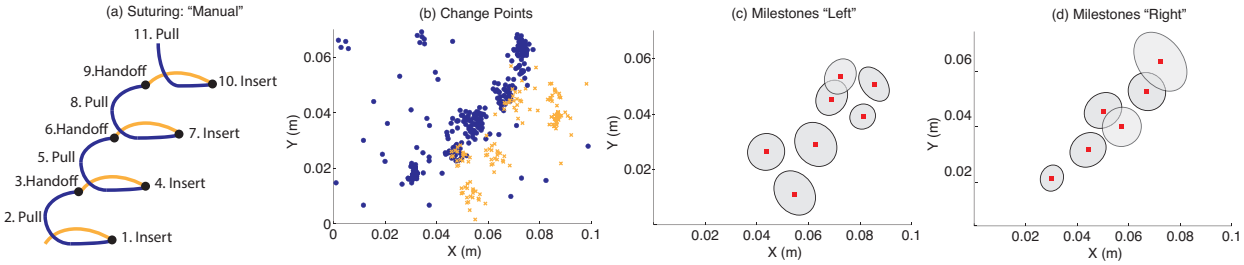

Fig. 7: (a) A manual semantic segmentation of suturing task. In orange, we mark the left arm motions and in blue we mark the right arm motions (b) The change points for the task are marked in orange (left arm) and blue (right arm). (c-d) The milestones, which are clusters of the change points, are illustrated with their 75% confidence ellipsoid for both arms

far more challenging than the previous tasks as the demonstrations were inconsistent. These inconsistency was in the way the suture is pulled after insertion (some pull to the left, some to the right, etc.), leading to change points all over the state space. Furthermore, there were numerous demonstrations with retry-until-success behaviors for the left arm. Our pruning and compaction techniques are able to mitigate some of this problems. 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 milestones clearly correspond to the manually segmented transitions. As the task progresses, we see that some of the later milestones do not. This is likely due to an error accumulation, where actions that were slightly different at the start became increasingly varied at the end.

6.4 Experiment 3. Comparison to "Surgemes"

In the previous experiments, we showed that our milestones qualitatively matched manually annotated segments. Our manual annotations highlighted only the most important steps in the task. Milestones give us a semantic task representation consistently recurring in data, which is in contrast to surgemes which segment at trajectory level. We evaluate how these primitives compare to the segmentation that we learn with milestones. While this is an apples-to-oranges comparison, it gives us some insights on how the two frameworks differ. The JIGSAWS dataset has annotations for every demonstration including the active surgeme type; the time at which the surgeme begins, and ends. It uses a dictionary of 15 surgemes.

In Table 1, we compare the number of milestone segments for needle passing and suturing to the number of annotated surgeme segments. We average the number of surgemes segments per demonstration and also show one standard deviation. We find that for both tasks there are less segments using milestones than there are using surgemes. A key difference between our segmentation and number of annotated surgemes is our compaction step. To account for this, we also apply a compaction step to the surgeme segments. While there can be other retrials, we manually identify two surgemes, needle repositioning and needle orientation, that frequently correspond to retrial steps in [5]. In case of consecutive appearances of these surgemes, we only keep the 1 instance of each for compaction.

Table 1: Since one-to-one comparison of learned milestones with annotated data is not meaningful, we compare our method with annotated data in terms of *coverage*, i.e. milestones which have exactly one annotated transition (after compaction) within its time interval.

	Mean no. of Surgeme Segments in Annotated Data	Mean no. of Surgeme Segments in Data (after Compaction)	No. of Milestones (Our Method)	Covered Milestones
Needle Passing	19.3 ± 3.2	14.4 ± 2.57	11	74%
Suturing	20.3 ± 3.5	15.9 ± 3.11	13	66%

The natural next question to explore is whether the milestones that we found are similar to any of the surgeme transitions. For each demonstration, we count the number of milestones that are *covered*, that is, there is only one surgeme (after compaction) that falls within the milestone's time interval. For the Needle Passing task, on average 74% of the milestones were covered, that means there was exactly 1 surgeme transition that corresponded to that milestone. However, for the Suturing task, on average, 66% of the milestones were covered. In terms of coverage, the compaction step is crucial. Without compaction, the suturing task has a coverage of only 46% due to the frequent retrials.

6.5 Experiment 4. Pruning and Compaction

In Figure 8, we highlight the benefit of pruning and compaction using the Suturing task as exemplar. First, we show the change points without applying the compaction step to remove retrial change points (Figure 8a). We find that there are many more change points at the "insert" step of the task. Compaction removes the segments that correspond to retrial of the insertions. Next, we show the all of the clusters found by DP-GMM. The centroids of these clusters are marked in Figure 8b. Many of these clusters are small containing only a few change points. This is why we created the heuristic to prune clusters that do not have change points from at least 80% of the demonstrations. In all, 11 clusters are pruned by this rule.

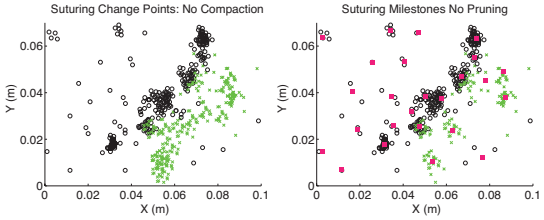

Fig. 8: We highlight the benefits of pruning and compaction. We first show the change points without compaction (in black and green), and then show the milestones without pruning (in red). Compaction sparsifies the change points and pruning significantly reduces the number of clusters.

7 Conclusion and Future Work

To summarize, we have presented an unsupervised semantic segmentation of surgical tasks using a novel framework that we call "milestones". Milestones are ellipsoidal regions of the state-space (constructed via poses and features) where demonstrations switch between different locally linear transition regimes. Milestones give a task-level segmentation, as opposed to segmenting individual trajectories, that divides a task into smaller logical phases. To find milestones, we proposed a hierarchical model that used Bayesian Non-Parametric clustering that leverages implicit information from both kinematic data as well as environment features such as visual cues. Our

results qualitatively matched hand-annotations of these tasks, and suggests significant improvements over alternative supervised techniques for segmentation.

Our limited featurization of only penetration and grasping events was sufficient to capture many of the important interactions. While, the focus of this study was evaluation of the hierarchical framework for milestone learning, The authors will explore autonomous visual feature generation. We believe that a richer and a more general featurization can help address some of the problems seen in our framework. For example, if use of needle pose in $z(t)$ may improve segmentation in suturing tasks. The key question is leveraging local structure in this complex new high-dimensional feature space. We believe that using a Radial Basis Function (RBF) kernel on the tuple $(\mathbf{x}(t), \mathbf{x}(t+1))$ might allow us to detect transitions in a kernelized space giving us additional robustness to non-linearities.

Acknowledgments: This research at UC Berkeley's Automation Sciences Lab was supported in part by a seed grant from the UC Berkeley Center for Information Technology in the Interest of Society (CITRIS), by the U.S. National Science Foundation under Award IIS-1227536: Multilateral Manipulation by Human-Robot Collaborative Systems. This work has been supported in part by funding from Google and and CISCO. We thank our colleagues who provided helpful feedback and suggestions, in particular Jeff Mahler and Michael Laskey for draft review.

References

[1] Antsaklis, P.J., Stiver, J.A., Lemmon, M.: Hybrid system modeling and autonomous control systems. In: Hybrid Systems, pp. 366–392. Springer (1993)

[2] Asfour, T., Gyarfas, F., Azad, P., Dillmann, R.: Imitation learning of dual-arm manipulation tasks in humanoid robots. In: Humanoid Robots, 2006 6th IEEE-RAS International Conference on, pp. 40–47 (2006)

[3] Calinon, S., Billard, A.: Stochastic gesture production and recognition model for a humanoid robot. In: Intelligent Robots and Systems, 2004. (IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on, vol. 3, pp. 2769–2774 vol.3 (2004)

[4] Elhamifar, E., Vidal, R.: Sparse subspace clustering. In: Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on, pp. 2790–2797. IEEE (2009)

[5] Gao, Y., Vedula, S., Reiley, C., Ahmidi, N., Varadarajan, B., Lin, H., Tao, L., Zappella, L., Bejar, B., Yuh, D., Chen, C., Vidal, R., Khudanpur, S., Hager, G.: 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) (2014)

[6] Grundmann, M., Kwatra, V., Han, M., Essa, I.: Efficient hierarchical graph-based video segmentation. In: Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on, pp. 2141–2148. IEEE (2010)

[7] He, J., Balzano, L., Szlam, A.: Incremental gradient on the grassmannian for online foreground and background separation in subsampled video. In: Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on, pp. 1568–1575. IEEE (2012)

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

[9] Kehoe, B., Kahn, G., Mahler, J., Kim, J., Lee, A., Lee, A., Nakagawa, K., Patil, S., Boyd, W., Abbeel, P., Goldberg, K.: Autonomous multilateral debridement with the raven surgical robot. In: Int. Conf. on Robotics and Automation (ICRA) (2014)

[10] Konidaris, G., Kuindersma, S., Grupen, R., Barto, A.: Robot learning from demonstration by constructing skill trees. The International Journal of Robotics Research (2011)

[11] Koprinska, I., Carrato, S.: Temporal video segmentation: A survey. Signal processing: Image communication **16**(5), 477–500 (2001)

[12] Kruger, V., Herzog, D., Baby, S., Ude, A., Kragic, D.: Learning actions from observations. Robotics & Automation Magazine, IEEE **17**(2), 30–43 (2010)
[13] Kurihara, K., Welling, M., Vlassis, N.A.: Accelerated variational dirichlet process mixtures. In: Advances in Neural Information Processing Systems, pp. 761–768 (2006)
[14] Lauer, F., Bloch, G.: Switched and piecewise nonlinear hybrid system identification. In: Hybrid Systems: Computation and Control, pp. 330–343. Springer (2008)
[15] Lea, C., Hager, G.D., Vidal, R.: An improved model for segmentation and recognition of fine-grained activities with application to surgical training tasks. In: WACV (2015)
[16] Lin, H., Shafran, I., Murphy, T., Okamura, A., Yuh, D., Hager, G.: Automatic detection and segmentation of robot-assisted surgical motions. In: Medical Image Computing and Computer-Assisted Intervention (MICCAI), pp. 802–810. Springer (2005)
[17] Mahler, J., Krishnan, S., Laskey, M., Sen, S., Murali, A., Kehoe, B., Patil, S., Wang, J., Franklin, M., Abbeel, P., K., G.: Learning accurate kinematic control of cable-driven surgical robots using data cleaning and gaussian process regression. In: Int. Conf. on Automated Sciences and Engineering (CASE), pp. 532–539 (2014)
[18] Murali, A., Sen, S., Kehoe, B., Garg, A., McFarland, S., Patil, S., Boyd, W., Lim, S., Abbeel, P., Goldberg, K.: Learning by observation for surgical subtasks: Multilateral cutting of 3d viscoelastic and 2d orthotropic tissue phantoms. In: Int. Conf. on Robotics and Automation (ICRA) (2015)
[19] Nicolescu, M.N., Mataric, M.J.: Natural methods for robot task learning: Instructive demonstrations, generalization and practice. In: Proceedings of the Second International Joint Conference on Autonomous Agents and Multiagent Systems, AAMAS '03, pp. 241–248. ACM, New York, NY, USA (2003). DOI 10.1145/860575.860614
[20] Niekum, S., Osentoski, S., Konidaris, G., Barto, A.: Learning and generalization of complex tasks from unstructured demonstrations. In: Int. Conf. on Intelligent Robots and Systems (IROS), pp. 5239–5246. IEEE (2012)
[21] Niekum, S., Osentoski, S., Konidaris, G., Chitta, S., Marthi, B., Barto, A.G.: Learning grounded finite-state representations from unstructured demonstrations. I. J. Robotic Res. **34**(2), 131–157 (2015). DOI 10.1177/0278364914554471
[22] Pastor, P., Hoffmann, H., Asfour, T., Schaal, S.: Learning and generalization of motor skills by learning from demonstration. In: Int. Conf. on Robotics and Automation (ICRA), pp. 763–768. IEEE (2009)
[23] Quellec, G., Lamard, M., Cochener, B., Cazuguel, G.: Real-time segmentation and recognition of surgical tasks in cataract surgery videos. Medical Imaging, IEEE Transactions on **33**(12), 2352–2360 (2014)
[24] Rosen, J., Brown, J.D., Chang, L., Sinanan, M.N., Hannaford, B.: Generalized approach for modeling minimally invasive surgery as a stochastic process using a discrete markov model. Biomedical Engineering, IEEE Transactions on **53**(3), 399–413 (2006)
[25] Tao, L., Zappella, L., Hager, G.D., Vidal, R.: Surgical gesture segmentation and recognition. In: Medical Image Computing and Computer-Assisted Intervention–MICCAI 2013, pp. 339–346. Springer (2013)
[26] Vakanski, A., Mantegh, I., Irish, A., Janabi-Sharifi, F.: Trajectory learning for robot programming by demonstration using hidden markov model and dynamic time warping. Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on **42**(4), 1039–1052 (2012)
[27] Varadarajan, B., Reiley, C., Lin, H., Khudanpur, S., Hager, G.: Data-derived models for segmentation with application to surgical assessment and training. In: Medical Image Computing and Computer-Assisted Intervention (MICCAI), pp. 426–434. Springer (2009)
[28] Willsky, A.S., Sudderth, E.B., Jordan, M.I., Fox, E.B.: Sharing features among dynamical systems with beta processes. In: Advances in Neural Information Processing Systems, pp. 549–557 (2009)
[29] Wolff, E.M., Murray, R.M.: Optimal control of nonlinear systems with temporal logic specifications. In: Proc. of the International Symposium on Robotics Research (ISRR) (2013)
[30] Zappella, L., Bejar, B., Hager, G., Vidal, R.: Surgical gesture classification from video and kinematic data. Medical image analysis **17**(7), 732–745 (2013)